JBoss Enterprise Application Platform 5
Manual del usuario de JBoss Messaging

---

para uso con JBoss Enterprise Application Platform 5
Edición 5.1.0

Tim Fox
Clebert Suconic
Jeff Mesnil
Howard Gao
Andy Taylor

JBoss Enterprise Application Platform 5 Manual del usuario de JBoss Messaging

---

para uso con JBoss Enterprise Application Platform 5
Edición 5.1.0

Tim Fox

Jeff Mesnil

Andy Taylor

Clebert Suconic

Howard Gao

Edited by

Laura Bailey

Jared Morgan

Legal Notice

Copyright © 2011 Red Hat, Inc.

This document is licensed by Red Hat under the Creative Commons Attribution-ShareAlike 3.0 Unported License. If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, MetaMatrix, Fedora, the Infinity Logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack Logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

Un manual del uso de JBoss Messaging 1.4 con JBoss Enterprise Application Platform 5.0 y sus lanzamientos de parches.

Table of Contents

Prefacio **4**
1. Convenciones del Documento 4
1.1. Convenciones tipográficas 4
1.2. Convenciones del documento 5
1.3. Notas y Advertencias 6
2. Cómo obtener ayuda y hacer sus comentarios 6
2.1. ¿Necesita ayuda? 6
2.2. ¡Necesitamos sus comentarios! 7

Capítulo 1. Sobre JBoss Messaging 1.4 . . . . . **8**

Capítulo 2. Introducción **9**
2.1. Características de JBoss Messaging 9
2.2. Compatibilidad con JBoss MQ 10
2.3. Propiedades del sistema que JBoss Messaging utiliza 10
2.3.1. support.bytesId 10
2.3.2. retain.oldxabehaviour 11

Capítulo 3. Instalación de JBoss Messaging **12**

Capítulo 4. Ejecución de los ejemplos **13**

Capítulo 5. Configuración **15**
5.1. Configuración del ServerPeer 15
5.2. Atributos del ServerPeer 17
5.2.1. Métodos del ServerPeer 20
5.3. Cambio de la base de datos 22
5.4. Configuración de la oficina postal 23
5.4.1. Atributos de la oficina postal 26
5.5. Configuración del administrador de persistencia 28
5.5.1. Atributos MBean PersistenceManager 30
5.6. Configuración del administrador de usuarios de JMS 31
5.6.1. Atributos Bean administrados JMSUserManager 32
5.7. Configuración de destinos 32
5.7.1. Destinos pre-configurados 32
5.7.2. Configuración de colas 33
5.7.2.1. Atributos MBean cola 33
5.7.2.1.1. Configuración de seguridad del destino 35
5.7.2.1.2. Parámetros de paginación de destinos 35
5.7.2.1.3. Operaciones Bean administradas en cola 36
5.7.3. Configuración de temas 37
5.7.3.1. Atributos Bean administrados por temas 37
5.7.3.1.1. Configuración de seguridad del destino 38
5.7.3.1.2. Parámetros de paginación de destinos 39
5.7.3.2. Operaciones Bean administradas por tema 40
5.8. Configuración de fábricas de conexión 40
5.8.1. Atributos del Bean administrado ConnectionFactory 43
5.9. Configuración del conector remoto 45
5.10. ServiceBindingManager 50

Capítulo 6. Notas sobre clústers **51**
6.1. Id único del servidor compañero 51
6.2. Destinos con clústers 51

6.3. Subscripciones durables en clústers 51
6.4. Destinos temporales en clústers 51
6.5. Servidores sin clústers 51
6.6. Ordenamiento de mensajes en el clúster 52
6.7. Operaciones Idempotent 52
6.8. Fábricas de conexiones en clústers 52

Capítulo 7. Configuración de JBoss Messaging XA Recovery 53

Capítulo 8. Configuración del puente de mensajes de JBoss Messaging 54
8.1. Sinopsis del puente de mensajes 54
8.2. Despliegue del puente 55
8.3. Configuración del puente 55

Capítulo 9. Habilitación del grupo de ordenamiento de JBoss Messaging 61
9.1. Cómo habilitar el grupo de ordenamiento de mensajes 61
9.1.1. Habilitación con el API 61
9.1.2. Cambios en la configuración 62
9.2. Notas y limitaciones 63

Historial de revisiones 64

Prefacio

1. Convenciones del Documento

Este manual utiliza varias convenciones para resaltar algunas palabras y frases y llamar la atención sobre ciertas partes específicas de información.

En ediciones PDF y de papel, este manual utiliza tipos de letra procedentes de Liberation Fonts. Liberation Fonts también se utilizan en ediciones de HTML si están instalados en su sistema. Si no, se muestran tipografías alternativas pero equivalentes. Nota: Red Hat Enterprise Linux 5 y siguientes incluyen Liberation Fonts predeterminadas.

1.1. Convenciones tipográficas

Se utilizan cuatro convenciones tipográficas para llamar la atención sobre palabras o frases específicas. Dichas convenciones y las circunstancias en que se aplican son las siguientes:

`Negrita monoespaciado`

Utilizado para resaltar la entrada del sistema, incluyendo los comandos de shell, nombres de archivos y rutas. También sirve para resaltar teclas y combinaciones de teclas. Por ejemplo:

> Para ver el contenido del archivo **`my_next_bestselling_novel`** en su directorio actual de trabajo, escriba el comando **`cat my_next_bestselling_novel`** en el intérprete de comandos de shell y pulse **`Enter`** para ejecutar el comando.

El ejemplo anterior incluye un nombre de archivo, un comando de shell y una tecla . Todo se presenta en negrita-monoespaciado y distinguible gracias al contexto.

Las combinaciones de teclas se pueden distinguir de las individuales con el signo más que conecta cada partee de la combinación de tecla. Por ejemplo:

> Pulse **`Enter`** para ejecutar el comando.
>
> Pulse **`Ctrl`**+**`Alt`**+**`F2`** para pasar a una terminal virtual.

El primer ejemplo resalta una tecla particular a pulsar. El segundo ejemplo, resalta una combinación de teclas: un set de tres teclas pulsadas simultáneamente.

Si se discute el código fuente, los nombres de las clase, los métodos, las funciones, los nombres de variables y valores de retorno mencionados dentro de un párrafo serán presentados en **`Negrita-monoespaciado`**. Por ejemplo:

> Las clases de archivo relacionadas incluyen **`filename`** para sistema de archivos, **`file`** para archivos y **`dir`** para directorios. Cada clase tiene su propio conjunto asociado de permisos.

Negrita proporcional

Esta denota palabras o frases encontradas en un sistema, incluyendo nombres de aplicación, texto de cuadro de diálogo, botones etiquetados, etiquetas de cajilla de verificación y botón de radio; títulos de menú y títulos del submenú. Por ejemplo:

> Seleccione **Sistema** → **Preferencias** → **Ratón** desde la barra del menú principal para lanzar **Preferencias de ratón**. En la pestaña de **`Botones`**, seleccione la cajilla de **`ratón de mano izquierda`** y luego haga clic en **`Cerrar`** para cambiar el botón principal del ratón de la izquierda a la derecha (adecuando el ratón para la mano izquierda).
>
> Para insertar un carácter especial en un archivo **`gedit`**, seleccione **Aplicaciones** → **Accesorios** → **Mapa de caracteres** de la barra del menú. Luego, seleccione **Búsqueda** → **Buscar...** de la barra del menú de **Mapa de caracteres**, escriba el nombre del carácter en el campo de **`Búsqueda`** y haga clic en **`Siguiente`**. El carácter que buscó será

resaltado en la **Tabla de caracteres**. Haga doble clic en ese carácter resaltado para colocarlo en el campo de **Texto a copiar** y luego haga clic en el botón **Copiar**. Ahora regrese al documento y elija **Modificar** → **Pegar** de la barra de menú de **gedit**.

El texto anterior incluye nombres de aplicación; nombres y elementos del menú de todo el sistema; nombres de menú de aplicaciones específicas y botones y texto hallados dentro de una interfaz gráfica de usuario, todos presentados en negrita proporcional y distinguibles por contexto.

Itálicas-negrita monoespaciado o ***Itálicas-negrita proporcional***

Ya sea negrita monoespaciado o negrita proporcional, la adición de itálicas indica texto reemplazable o variable. Las itálicas denotan texto que usted no escribe literalmente o texto mostrado que cambia dependiendo de la circunstancia. Por ejemplo:

Para conectar a una máquina remota utilizando ssh, teclee **ssh *nombre de usuario*@*dominio.nombre*** en un intérprete de comandos de shell. Si la máquina remota es **example.com** y su nombre de usuario en esa máquina es john, teclee **ssh john@example.com**.

El comando **mount -o remount *file-system*** remonta el sistema de archivo llamado. Por ejemplo, para volver a montar el sistema de archivo **/home**, el comando es **mount -o remount /home**.

Para ver la versión de un paquete actualmente instalado, utilice el comando **rpm -q *paquete***. Éste entregará el resultado siguiente: ***paquete-versión-lanzamiento***.

Observe que las palabras resaltadas en itálicas — nombre de usuario, dominio.nombre, sistema de archivo, paquete, versión y lanzamiento. Cada palabra es un marcador de posición, ya sea de texto a ingresar cuando se ejecuta un comando o para un texto ejecutado por el sistema.

Aparte del uso estándar para presentar el título de un trabajo, las itálicas denotan el primer uso de un término nuevo e importante. Por ejemplo:

Publican es un sistema de publicación de *DocBook*.

1.2. Convenciones del documento

Los mensajes de salida de la terminal o fragmentos de código fuente se distinguen visualmente del texto circundante.

Los mensajes de salida enviados a una terminal se muestran en **romano monoespaciado** y se presentan así:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Los listados de código fuente también se muestran en **romano monoespaciado**, pero se presentan y resaltan de la siguiente manera:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }
}
```

1.3. Notas y Advertencias

Finalmente, utilizamos tres estilos visuales para llamar la atención sobre la información que de otro modo se podría pasar por alto.

Nota

Una nota es una sugerencia, atajo o enfoque alternativo para una tarea determinada. Ignorar una nota no debería tener consecuencias negativas, pero podría perderse de algunos trucos que pueden facilitarle las cosas.

Importante

Los cuadros con el título de importante dan detalles de cosas que se pueden pasar por alto fácilmente: cambios de configuración únicamente aplicables a la sesión actual, o servicios que necesitan reiniciarse antes de que se aplique una actualización. Ignorar estos cuadros no ocasionará pérdida de datos, pero puede causar enfado y frustración.

Aviso

Las advertencias no deben ignorarse. Ignorarlas muy probablemente ocasionará pérdida de datos.

2. Cómo obtener ayuda y hacer sus comentarios

2.1. ¿Necesita ayuda?

Si encuentra dificultades con alguno de los procedimientos descritos en este documento, visite el Portal del cliente de Red Hat en http://access.redhat.com. A través del portal del cliente, usted podrá:

- buscar o navegar a través de la base de artículos de soporte técnico sobre productos de Red Hat.
- enviar un caso de soporte a Servicios de Soporte Global de Red Hat (GSS)
- acceder a otra documentación del producto.

Red Hat alberga una lista grande de correos electrónicos para discutir sobre software de Red Hat y tecnología. Encontrará un listado de las listas de correo disponibles al público en

https://www.redhat.com/mailman/listinfo. Haga clic en el nombre de la lista a la que quiera suscribirse o para acceder a los archivos de listados.

2.2. ¡Necesitamos sus comentarios!

Si encuentra algun error o si se le ocurre una manera de mejorar este manual, nos encantaría escuchar sus sugerencias. Complete un reporte en Bugzilla frente al producto **JBoss Enterprise Application Platform 5** y el componente **doc-JBoss_Messaging_User_Guide**. El siguiente enlace le llevará a un reporte de error ya completado para este producto: http://bugzilla.redhat.com/.

Llene la siguiente plantilla en el campo de **Description** de Bugzilla. Sea tan especifico como le sea posible al describir el problema, esto ayudará a asegurarnos de que lo podemos solucionar rápidamente.

```
URL del documento:

Número de la sección y nombre:

Describa el problema:

Sugerencias para mejorar:

Información adicional:
```

Asegúrese de darnos su nombre para poder darle todo el crédito por reportar el problema.

Capítulo 1. Sobre JBoss Messaging 1.4

JBoss Messaging es un sistema de mensajería empresarial de JBoss. Es una completa re-escritura de JBossMQ, el proveedor JMS del servicio de mensajería Java JBoss.

JBoss Messaging es el proveedor JMS predeterminado en JBoss Enterprise Application Platform 4.3 y 5.

JBoss Messaging es una parte integral de la estrategia de mensajería de Red Hat. Brinda un rendimiento mejorado en entornos con un sólo nodo o con clústers. También cuenta con una arquitectura modular que permite el añadir fácilmente nuevas funcionalidades en el futuro.

Este manual muestra cómo instalar y configurar JBoss Messaging para JBoss Enteprise Application Platform.

Capítulo 2. Introducción

JBoss Messaging proporciona una plataforma de mensajería de código abierto y con base en los estándares para traer la mensajería de clase empresarial al mercado en masa.

JBoss Messaging implementa un núcleo de mensajes de alto rendimiento y robusto que está diseñado para soportar la arquitectura orientada a servicios (en inglés Service Oriented Architectures o SOAs), ESBs (del inglés Enterprise Service Buses) y otras requerimientos de integración sin importar el nivel de demanda.

JBoss Messaging le permite distribuir la carga de la aplicación de manera equitativa a través del clúster. Balancea los ciclos de la CPU de cada nodo sin ningún punto de fallo, proporcionando una implementación en clúster altamente escalable y con un alto rendimiento.

JBoss Messaging incluye un servicio de mensajería Java - JMS del inglés Java Messaging Service de manera que los mensajes se entregan en un formato basado en los estándares y para habilitar el soporte para otros protocolos de mensajería en el futuro.

2.1. Características de JBoss Messaging

JBoss Messaging proporciona las siguientes funcionalidades:

- Un fuerte enfoque en el rendimiento, confiabilidad y escalabilidad con un alto rendimiento y baja latencia.
- Una base para JBoss ESB para las iniciativas SOA, (JBoss ESB usa JBoss Messaging como su proveedor JMS predeterminado).

JBoss Messaging también incluye:

- Modelos de mensajería punto-a-punto y publicar-suscribir;
- Mensajes persistentes y no-persistentes;
- Entrega garantizada de mensajes lo que asegura que los mensajes llegan sólo una vez a donde se requiere;
- Interfaz transaccional y confiable que soporta la semántica ACID;
- Un marco de trabajo de seguridad personalizable basado en JAAS;
- Integración completa con JBoss Transactions (conocido antes como Arjuna JTA) para soportar la recuperación completa de transacciones;
- Una interfaz extensiva de administración JMX;
- Soporte para la mayoría de las bases de datos más importantes incluyendo Oracle, DB2, Sybase, Microsoft SQL Server, PostgreSQL y MySQL;
- Transporte HTTP para permitir el uso a través de cortafuegos que sólo permiten tráfico HTTP;
- Transporte servlet para permitir la mensajería por medio de un servlet dedicado;
- Transporte SSL;
- DLQs (del inglés Dead Letter Queues) configurables y colas de vencimiento;
- Estadísticas de mensajes, las cuales proporcionan una vista histórica de los mensajes que se entregaron a qué colas y a qué subscripciones;
- La paginación automática de los mensajes al almacenamiento. Esto le permite el uso de colas muy largas que serían demasiado largas para caber en la memoria del sistema por completo;
- Ordenamiento estricto de mensajes, lo cual hace que los mensajes que pertenecen a un grupo de mensajes en particular sean entregados de acuerdo al orden en que llegaron a la cola de destino.

JBoss Messaging también incluye las siguientes funcionalidades de clústers:

- **Colas y temas completamente en clústers**

  Los temas y colas *lógicas* se distribuyen a través del clúster. Puede enviar o recibir una cola o un tema desde y hacia cualquier nodo en el clúster.

- **Subscripciones durables completamente en clústers**

Una subscripción durable en particular se puede acceder desde cualquier nodo del clúster, lo cual le permite distribuir la carga de procesamiento desde esa subscripción a través de todo el clúster.

- **Colas temporales completamente en clústers**

  Si un mensaje enviado incluye el **`replyTo`** de una cola temporal se puede retornar en cualquier nodo del clúster.

- **Redistribución inteligente de mensajes**

  Los mensajes se mueven de manera automática entre nodos del clúster para tomar ventaja de las diferentes velocidades del consumidor en diferentes nodos. Esto ayuda a prevenir el hambre o la acumulación de mensajes en un nodo en particular.

- **Protección del orden de mensajes**

  Habilite esto para asegurarse de que el orden de los mensajes generados por un productor es idéntico al orden de los mensajes consumidos por un consumidor. Esto funciona incluso si la redistribución de mensajes está activa.

- **Conmutación de servidores en caso de fallo de manera completamente transparente**

  Cuando un servidor falla, sus sesiones continuan sin excepciones en un nuevo nodo. Esto es completamente configurable: si no quiere implementar este comportamiento de conmutación de servidores en caso de fallo, puede desactivarlo y regresar a las excepciones y manualmente recrear las conexiones en un nuevo nodo.

- **Alta disponibilidad y conmutación no aparente del servidor en caso de fallos**

  Si el nodo falla, conmutará de manera automática a otro nodo y no perderá ningún mensaje persistente y puede continuar con su sesión sin ningún cambio aparente. En todo momento se respeta la política de entrega de mensajes de *sólo una vez*.

- **Puente de mensajes**

  JBoss Messaging contiene un componente de puente de mensajes, el cual le habilita a establecer puentes de mensajes entre dos destinos cualesquiera JMS1.1. Esto le permite conectar clústers separados geográficamente, formando grandes temas y colas lógicas globalmente distribuídas.

2.2. Compatibilidad con JBoss MQ

JBoss MQ era la implementación de JMS enviada con Enterprise Application Platform 4.2. Ya que JBoss Messaging es compatible con JMS 1.1 y JMS 1.0.2b, el código JMS escrito frente a JBoss MQ ejecutará con JBoss Messaging sin ningún cambio.

JBoss Messaging no tiene compatibilidad de formato con JBoss MQ así que es necesario actualizar los clientes JBoss MQ con JARs de clientes JBoss Messaging.

Importante

Aunque los descriptores de despliegue de JBoss Messaging son similares a los descriptores de despliegue de JBoss MQ, *no son idénticos* así que necesitarán algunos ajustes simples para que funcionen con JBoss Messaging. El modelo de datos de la base de datos es completamente diferente así que no intente utilizar JBoss Messaging con un esquema de datos JBoss MQ y vice-versa.

2.3. Propiedades del sistema que JBoss Messaging utiliza

2.3.1. support.bytesId

Esta propiedad del sistema controla el comportamiento predeterminado al construir un objeto **JBossMessage** desde un objeto foráneo de mensajes. Establezca esta propiedad por medio de la línea de comandos durante el arranque del servidor con la opción **-D**.

Si esta propiedad se configura como **true** entonces el constructor **JBossMessage** tratará de extraer el ID de correlación byte[] nativo de los encabezados foráneos del mensaje. Si se configura como **false**, utilizará el tipo de cadena normal **JMSCorrelationID**. Esta propiedad por defecto es **true** si no se establece o cuando se configura con algo diferente a **true** o **false**.

2.3.2. retain.oldxabehaviour

Esta propiedad del sistema controla el tipo de excepción que presenta un JMS XAResource en caso de que se llame el método **prepare()** después de que se rompa la conexión. Establezca esta propiedad por medio de la línea de comandos durante el arranque del servidor con la opción **-D**.

Si esta propiedad no se define entonces se presenta una **XAException** con un código de error **XA_RBCOMMFAIL**. De otra manera se presenta una **XAException** con un código de error **XA_RETRY**. Debe notar que JBoss Messaging no define esta propiedad por defecto.

Capítulo 3. Instalación de JBoss Messaging

JBoss Enterprise Application Platform (EAP) viene junto con JBoss Messaging pre-instalado como proveedor JMS predeterminado. Si está utilizando EAP versión 4.3 o posterior no ha necesidad de instalar manualmente JBoss Messaging.

Capítulo 4. Ejecución de los ejemplos

JBoss Messaging cuenta con un número de ejemplos que están disponibles para descargar. Consulte el *Installation Guide* que se envía junto con JBoss Enterprise Application Platform para aprender a instalar y configurar los ejemplos. El *Installation Guide* está disponible en http://docs.redhat.com/ bajo *JBoss Enterprise Application Platform > 5.0*.

Descomprima el archivo de ejemplos en el directorio **$JBOSS_HOME/docs/examples**.

Antes de ejecutar estos ejemplos, implemente el **jbm-examples-destinations-service** ubicado bajo **$JBOSS_HOME/docs/examples/jboss-messaging-examples/destinations/** para **$JBOSS_HOME/server/default/deploy**. El

Los siguientes ejemplos están incluídos con JBoss Messaging:

$JBOSS_HOME/docs/examples/jboss-messaging-examples/queue/

Este ejemplo demuestra un simple envío y recepción a una cola remota utilizando un cliente JMS.

$JBOSS_HOME/docs/examples/jboss-messaging-examples/queue-failover/

Este ejemplo demuestra la conmutación transparente de un consumidor JMS.

$JBOSS_HOME/docs/examples/jboss-messaging-examples/topic/

Este ejemplo demuestra un simple enviar y recibir a un tema remoto utilizando un cliente JMS.

$JBOSS_HOME/docs/examples/jboss-messaging-examples/mdb/

Este ejemplo demuestra el uso de un Enterprise JavaBean 2.1 MDB con JBoss Messaging.

$JBOSS_HOME/docs/examples/jboss-messaging-examples/ejb3mdb/

Este ejemplo demuestra el uso de un Enterprise JavaBean 3.0 MDB con JBoss Messaging.

$JBOSS_HOME/docs/examples/jboss-messaging-examples/stateless/

Este ejemplo demuestra un bean de sesión sin estado Enterprise JavaBean 2.1 interactuando con JBoss Messaging.

$JBOSS_HOME/docs/examples/jboss-messaging-examples/mdb-failure/

Este ejemplo demuestra la manera de deshacer y re-entregar dentro de un Enterprise JavaBean 2.1.

$JBOSS_HOME/docs/examples/jboss-messaging-examples/secure-socket/

Este ejemplo demuestra un cliente JMS interactuando con un servidor JBoss Messaging por medio de un transporte encriptado SSL.

$JBOSS_HOME/docs/examples/jboss-messaging-examples/http/

Este ejemplo demuestra un cliente JMS interactuando con un servidor JBoss Messaging construyendo un tunel para el tráfico a través de HTTP.

$JBOSS_HOME/docs/examples/jboss-messaging-examples/web-service/

Este ejemplo demuestra JBoss Webservice interactuando con JBoss Messaging.

$JBOSS_HOME/docs/examples/jboss-messaging-examples/distributed-queue/

Este ejemplo demuestra el cliente JMS interactuando con una cola distribuída de JBoss Messaging. Para ejecutar este ejemplo se requieren dos instancias de JBoss Enterprise

Application Platform.

$JBOSS_HOME/docs/examples/jboss-messaging-examples/distributed-topic/

Este ejemplo demuestra un cliente JMS interactuando con un tema distribuído de JBoss Messaging. Para ejecutar este ejemplo se requiere dos instancias de JBoss Enterprise Application Platform se encuentren ejecutando.

$JBOSS_HOME/docs/examples/jboss-messaging-examples/servlet/

Este ejemplo demuestra cómo utilizar el transporte servlet con JBoss Messaging. Implementa un servlet y una **ConnectionFactory**, la cual utiliza el transporte servlet.

$JBOSS_HOME/docs/examples/jboss-messaging-examples/ordering-group/

Este ejemplo demuestra el uso del ordenamiento estricto de mensajes. Usa la API del grupo de ordenamiento de JBoss Messaging para entregar mensajes estrictamente ordenados sin importar la prioridad de los mensajes.

$JBOSS_HOME/docs/examples/jboss-messaging-examples/bridge/

Este ejemplo demuestra el uso de un puente de mensajes. Despliega un puente de mensajes dentro de JBoss Enterprise Application Platform, el cual mueve los mensajes desde una cola fuente a una destino.

Familiarícese con los ejemplos para ganar una buena comprensión de JBoss Messaging.

Importante - Ejemplos sin clústers

Los ejemplos sin clústers esperan que una instancia de JBoss Enterprise Application Platform esté ejecutando con todos los valores predeterminados. El **readme.html** para cada ejemplo proporciona los detalles de configuración, la salida esperada y resolución de problemas simples.

Importante - Ejemplos en clústers

Los ejemplos con clústers requieren que dos instancias de JBoss Enterprise Application Platform estén ejecutando con nombres de clústersy puertos correctamente configurados. El **readme.html** para cada ejemplo proporciona los detalles de configuración, las salidas que se esperan y resolución de problemas simples.

Capítulo 5. Configuración

El API JMS (del inglés Java Message Service) especifica la manera en que un cliente de mensajería interactúa con un servidor de mensajería. La manera en que los servicios de mensajería tal como los destinos de mensajes y las fábricas de conexiones se definen y se implementan depende del proveedor JMS. JBoss Messaging tiene sus propios archivos para configuración de servicios.

Este capítulo le muestra la manera de configurar varios servicios disponibles en JBoss Messaging que funcionan juntos para proporcionar servicios a nivel de API JMS para las aplicaciones cliente.

La configuración de JBoss Messaging abarca varios archivos de configuración. Dependiendo de la funcionalidad que brindan los servicios que configura, los datos de configuración se distribuyen entre **`messaging-service.xml`**, **`remoting-bisocket-service.xml`**, **`xxx-persistence-service.xml (en donde xxx es el nombre de su base de datos)`**, **`connection-factories-service.xml`** y **`destinations-service.xml`**.

Las pilas del interceptor AOP se configuran en **`aop-messaging-client.xml`** (para el comportamiento del lado del cliente) y **`aop-messaging-server.xml`** (para el comportamiento del lado del servidor). Usualmente no hay necesidad de cambiar estos archivos pero algunos de los interceptores se pueden remover para mejorar el rendimiento en caso de que no los necesite. Antes de remover el interceptor de seguridad tenga en consideración las implicaciones en la seguridad.

5.1. Configuración del ServerPeer

El **`ServerPeer`** es el corazón de la fachada de JBoss Messaging JMS. Puede configurar su comportamiento modificando **`$JBOSS_HOME/server/$PROFILE/deploy/messaging/messaging-service.xml`**.

Todos los servicios de JBoss Messaging se basan en el **`ServerPeer`**.

A continuación encontrará un ejemplo de una configuración de un servidor compañero. Observe que no todos los valores para los atributos del servidor compañero se especifican en el ejemplo.

```
<!-- ServerPeer MBean configuration
     ============================== -->

<mbean code="org.jboss.jms.server.ServerPeer"
  name="jboss.messaging:service=ServerPeer"
  xmbean-dd="xmdesc/ServerPeer-xmbean.xml">

  <!--The unique id of the server peer - in a cluster each node
      MUST have a unique value - must be an integer-->

  <attribute name="ServerPeerID">
    ${jboss.messaging.ServerPeerID:0}
  </attribute>

  <!--The default JNDI context to use for queues when they are
      deployed without specifying one-->

  <attribute name="DefaultQueueJNDIContext">/queue</attribute>

  <!--The default JNDI context to use for topics when they are
      deployed without specifying one -->

  <attribute name="DefaultTopicJNDIContext">/topic</attribute>

  <attribute name="PostOffice">
    jboss.messaging:service=PostOffice
  </attribute>

  <!-- The default Dead Letter Queue (DLQ) to use for destinations.
       This can be overridden on a per destinatin basis -->

  <attribute name="DefaultDLQ">
    jboss.messaging.destination:service=Queue,name=DLQ
  </attribute>

  <!--The default maximum number of times to attempt delivery of a
      message before sending to the DLQ (if configured).
      This can be overridden on a per destination basis-->

  <attribute name="DefaultMaxDeliveryAttempts">10</attribute>

  <!--The default Expiry Queue to use for destinations. This can
      be overridden on a per destinatin basis-->

  <attribute name="DefaultExpiryQueue">
    jboss.messaging.destination:service=Queue,name=ExpiryQueue
  </attribute>

  <!--The default redelivery delay to impose. This can be overridden
      on a per destination basis -->

  <attribute name="DefaultRedeliveryDelay">0</attribute>

  <!--The periodicity of the message counter manager enquiring on
      queues for statistics-->

  <attribute name="MessageCounterSamplePeriod">5000</attribute>

  <!--The maximum amount of time for a client to wait for failover
      to start on the server side after it has detected failure-->

  <attribute name="FailoverStartTimeout">60000</attribute>

  <!--The maximum amount of time for a client to wait for failover
      to complete on the server side after it has detected failure-->

  <attribute name="FailoverCompleteTimeout">300000</attribute>
```

```
  <attribute name="StrictTck">false</attribute>

  <!--The maximum number of days results to maintain in the message
      counter history-->

  <attribute name="DefaultMessageCounterHistoryDayLimit">-1</attribute>

  <!--The name of the connection factory to use for creating
      connections between nodes to pull messages-->

  <attribute name="ClusterPullConnectionFactoryName">
    jboss.messaging.connectionfactory:service=ClusterPullConnectionFactory
  </attribute>

  <!--When redistributing messages in the cluster. Do we need to
      preserve the order of messages received
      by a particular consumer from a particular producer? -->

  <attribute name="DefaultPreserveOrdering">false</attribute>

  <!-- Max. time to hold previously delivered messages back waiting for
       clients to reconnect after failover -->

  <attribute name="RecoverDeliveriesTimeout">300000</attribute>

  <!-- Set to true to enable message counters that can be viewed via JMX -->

  <attribute name="EnableMessageCounters">false</attribute>

  <!-- The password used by the message sucker connections to create connections.
       THIS SHOULD ALWAYS BE CHANGED AT INSTALL TIME TO SECURE SYSTEM
  <attribute name="SuckerPassword"></attribute>
  -->

  <!-- The name of the server aspects configuration resource
  <attribute name="ServerAopConfig">aop/jboss-aop-messaging-server.xml</attribute>
  -->
  <!-- The name of the client aspects configuration resource
    <attribute name="ClientAopConfig">aop/jboss-aop-messaging-
client.xml</attribute>
  -->

  <depends optional-attribute-name="PersistenceManager">
    jboss.messaging:service=PersistenceManager
  </depends>

  <depends optional-attribute-name="JMSUserManager">
    jboss.messaging:service=JMSUserManager
  </depends>

  <depends>jboss.messaging:service=Connector,transport=bisocket</depends>
  <depends optional-attribute-name="SecurityStore"
    proxy-type="org.jboss.jms.server.SecurityStore">
      jboss.messaging:service=SecurityStore
  </depends>
</mbean>
```

5.2. Atributos del ServerPeer

Esta sección aborda los atributos del bean administrado **`ServerPeer`**.

ServerPeerID

El identificador único del **`ServerPeer`**. Todo nodo implementado tiene que tener un identificador único ya sea que los nodos formen un clúster o que estén enlazados por medio de un puente de mensajes. El identificador debe ser un número entero válido.

DefaultQueueJNDIContext

El contexto JNDI predeterminado a utilizar al enlazar colas. Por defecto es **`/queue`**.

DefaultTopicJNDIContext

El contexto JNDI predeterminado al utilizar al enlazar temas. Por defecto es **`/topic`**.

PostOffice

Esta es la oficina postal que el **`ServerPeer`** utiliza. Usualmente no es necesario que cambie este atributo. La oficina postal es la responsable de enrutar los mensajes a colas y de mantener el mapeo entre direcciones y colas.

DefaultDLQ

El DLQ (Dead Letter Queue) predeterminado que el servidor utiliza para los destinos. El DLQ se puede sobreescribir destino por destino. Para mayor información consulte Sección 5.7, “Configuración de destinos”. Un DLQ es un destino a donde se envían los mensajes cuando el servidor ha intentado entregarlos sin tener éxito más de cierto número de veces. Si no se especifica el QLD entonces el mensaje se borrará después del número máximo de intentos de entrega. El número máximo de intentos de entrega se puede especificar utilizando el atributo **`DefaultMaxDeliveryAttempts`** para un valor predeterminado global o individualmente destino por destino.

DefaultMaxDeliveryAttempts

El número predeterminado y máximo de veces que se intentará entregar un mensaje antes de enviar el mensaje al DLQ, si está configurado. El valor predeterminado es **`10`**. puede sobreescribir este valor destino por destino.

DefaultExpiryQueue

La cola de expiración predeterminada que el **`ServerPeer`** utilizará para los destinos. Puede sobreescribir este valor destino por destino - como se puede ver en la sección sobre la configuración del bean administrado de destino. Una cola de expiración mantiene los mensajes que han expirado. La expiración de un mensaje está determinada por el valor de **`Message::getJMSExpiration()`**. Si la cola de expiración no se especifica entonces el mensaje se borrará después de que ha expirado.

DefaultRedeliveryDelay

Este atributo le permite retrasar un intento de re-entrega, lo cual ayuda a prevenir el acabar con la entrega-fallo. El valor predeterminado es **`0`** (es decir, no hay retraso). Puede sobreescribir este valor destino por destino.

MessageCounterSamplePeriod

Este atributo define el periodo de tiempo entre las peticiones del servidor a la cola en busca de estadísticas sobre la cola. El valor predeterminado es **`5000`** milisegundos.

FailoverStartTimeout

El número máximo de milisegundos que el cliente esperará para que inicie la conmutación en caso de fallo en el lado del servidor cuando se detecta un problema. El valor predeterminado es **`60000`** (un minuto).

FailoverCompleteTimeout

El número máximo de milisegundos que el cliente esperará para que complete la conmutación en caso de fallo en el lado del servidor después de que ha iniciado la conmutación de servidores. El valor predeterminado es **`300000`** (cinco minutos).

DefaultMessageCounterHistoryDayLimit

JBoss Messaging proporciona un historial de conteo de mensajes, el cual muestra el número de mensajes que llegan a cada cola durante cierto número de días. Este atributo representa el número máximo de días que se debe almacenar el historial de conteo de mensajes. Se puede sobreescribir destino por destino.

ClusterPullConnectionFactoryName

La fábrica de conexiones que se utiliza para traer o extraer los mensajes entre colas. Puede omitir este atributo para inhabilitar la extracción de mensajes y retener la conmutación de servidores en caso de fallos.

DefaultPreserveOrdering

Si es **`true`** entonces el ordenamiento JMS se preserva en el clúster. Consulte Capítulo 6, *Notas sobre clústers* para obtener mayores detalles. Por defecto es **`false`**.

RecoverDeliveriesTimeout

Cuando la conmutación en caso de fallo tiene lugar, los mensajes que ya se han entregado se almacenan en espera de que los clientes se reconecten. Dado el caso en que los clientes no se vuelvan a conectar (por ejemplo el cliente está muerto) entonces eventualmente estos mensajes expirarán y se añadirán de regreso a la cola. Este atributo establece el valor en milisegundos. Por defecto es **`300000`** (cinco minutos).

EnableMessageCounters

Cuando es **`true`** se activan los contadores de mensajes cuando el servidor inicia.

SuckerPassword

JBoss Messaging internamente crea conexiones entre nodos con el fin de redistribuir los mensajes entre destinos en clústers. Estas conexiones se crean con el nombre de usuario reservado. Este atributo define la contraseña a utilizar al crear estas conexiones.

Para las versiones de JBoss Messaging posteriores a 1.4.1.GA, debe definir la **`SuckerPassword`** en el **`SecurityMetadataStore`**.

Aviso

El **`SuckerPassword`** se tiene que cambiar en el momento de la instalación o de otra manera se utilizará la contraseña predeterminada. Cualquier persona que conozca la contraseña predeterminada podrá ganar acceso a cualquier destino en el servidor.

SuckerConnectionRetryTimes

Este es el número máximo de veces que se le permite reintentar a la conexión de un sucker en caso de un fallo. El valor predeterminado es **`-1`**, el cual representa "re-intente indefinidamente".\n\t\n

SuckerConnectionRetryInterval

Este es el intervalo en milisegundos entre cada intento de la conexión del sucker fallido. El valor predeterminado es **`5000`**.

StrictTck

Para habilitar la semántica TCK (del inglés Technology Compatibility Kit) JMS estricta, configure este atributo como **`true`**.

Destinos

Retorna una lista de los destinos (colas y temas) actualmente desplegados.

MessageCounters

Un contador de mensajes para una cola en particular.

MessageStatistics

Estadísticas sobre cada contador de mensajes para cada cola.

SupportsFailover

Cuabndo este atributo es **`false`**, la conmutación de servidores en caso de fallo del lado del servidor no ocurre cuando un nodo se cuelga en un clúster.

PersistenceManager

Este es el administrador de persistencia que el **`ServerPeer`** utiliza, (normalmente no es necesario que cambie este atributo).

JMSUserManager

El administrador del usuario JMS que el **`ServerPeer`** utiliza, (normalmente no es necesario cambiar este atributo).

SecurityStore

El **`SecurityStore`** enchufable. Si redefine este atributo, recuerde que necesitará autenticar el usuario **`MessageSucker`** (**`JBM.SUCKER`**) con todos los permisos especiales que los clústers requieren.

SupportsTxAge

Especifica si el tiempo de creación de la transacción se lamacena en el registro de la transacción. Si se configura como **`true`** entonces se almacena el registro de la transacción. El valor predeterminado es **`false`**.

5.2.1. Métodos del ServerPeer

Los siguientes métodos están disponibles para el bean administrado ServerPeer:

deployQueue

Utilizado de manera programática para implementar una cola. Si la cola ya existe pero no se ha implementado entonces se implementa. De otra manera se crea y se implementa.

El parámetro **`name`** coincide con el nombre del destino a desplegar.

El parámetro **`jndiName`** opcional representa el nombre JNDI completo a donde enlazar el destino. Si no se especifica entonces el destino se enlazará en **`<DefaultQueueJNDIContext>/<name>`**.

Hay dos versiones sobrecargadas de esta operación. La primera implementa el destino con los parámetros de paginación predeterminados. La segunda despliega el destino con los parámetros de paginación especificados. Para mayor información sobre los parámetros de paginación consulte Sección 5.7, “Configuración de destinos”.

undeployQueue

Utilizado programáticamente para borrar la implementación de una cola. Las colas no se borran

del almacenamiento persistente. Esta operación retorna **`true`** si la cola fue desimplementada exitósamente. De otra manera retorna **`false`**.

destroyQueue

Utilizado de manera programática para destruir una cola. La cola es desimplementada y luego todos sus datos se destruyen de la base de datos.

Aviso

Tenga cuidado al utilizar este método ya que borrará todos los datos para la cola.

Esta operación retorna **`true`** si la cola se destruyó éxitosamente. De otra manera retorna **`false`**.

deployTopic

Se utiliza de manera programática para implementar un tema. Hay dos versiones sobrecargadas de esta operación. La primera implementa los temas ya existentes con los parámetros de paginación predeterminados. La segunda crea y despliega temas con los parámetros de paginación especificados. Consulte Sección 5.7, “Configuración de destinos” para obtener mayor información.

El parámetro **`name`** representa el nombre del destino a implementar.

El parámetro **`jndiName`** representa el nombre JNDI completo de la ubicación a donde enlazar el destino. Si esto no se especifica entonces el destino se enlaza en **`<DefaultTopicJNDIContext>/<name>`**.

undeployTopic

Utilizado para borrar programáticamente la implementación de un tema. Se borra la implementación de los temas pero no se borran de un almacenamiento persistente. Esta operación retorna **`true`** si se borra la implementación del tema de manera exitosa. De otra manera, se retorna **`false`**.

destroyTopic

Utilizado para destruir programáticamente un tema. Se borra la implementación de los temas y todos los datos se borran de la base de datos y se destruyen. Esta operación retorna **`true`** si el tema fue destruído exitósamente. De otra forma retorna **`false`**.

Aviso

Tenga cuidado al utilizar este método ya que borrará todos los datos para el tema.

listMessageCountersHTML

Retorna contadores de mensajes en un formato HTML de fácil presentación.

ResetAllMesageCounters

Reestablece todos los contadores de mensajes a cero.

enableMessageCounters

Habilita todos los contadores de mensajes para todos los destinos. Los contadores de mensajes están desactivados por defecto.

disableMessageCounters

Inactiva todos los contadores de mensajes para todos los destinos. Los contadores de mensajes están desactivados por defecto.

retrievePreparedTransactions

Recupera una lista de los XIDs para todas las transacciones actualmente en un estado preparado en el nodo.

showPreparedTransactions

Recupera una lista de los XIDs para todas las transacciones actualmente en un estado preparado en el nodo en un formato HTML de fácil presentación.

listAllPreparedTransactions

Presenta los detalles de todas las transacciones preparadas.

listPreparedTransactions

Presenta los detalles de todas las transacciones preparadas en donde la edad de la transacción sea igual o mayor que el tiempo especificado.

showMessageDetails

Presenta los detalles de un mensaje. El ID del mensaje se utiliza para especificar el mensaje que se va a presentar.

commitPreparedTransaction

Guarda manualmente los cambios de una transacción preparada. El Id de la transacción se utiliza para especificar la transacción que se va a guardar.

rollbackPreparedTransaction

Deshace manualmente una transacción preparada. El ID de la transacción se utiliza para especificar la transacción que se va a deshacer.

Debe cambiar su base de datos.

La configuración predeterminada de persistencia funciona con Hypersonic (HSQLDB) de manera que las plataformas empresariales de JBoss Enterprise Platforms pueden ejecutar "tal como vienen empacadas". Sin embargo, *Hypersonic no se soporta en producción y no se debe ustilizar en un entorno de producción.*
Los problemas conocidos con la base de datos Hypersonic incluyen:

- No hay aislamiento de transacciones
- Fugas de sockest e hilos (**`connection.close()`** no limpia los recursos)\n\t\n
- Calidad de persistencia y (los registros usualmente se dañan después de un fallo, lo cual evita una recuperación automática)
- Corrupción de la base de datos
- Estabilidad bajo carga (los procesos de la base de datos se detienen cuando hay demasiados datos)
- No es viable en entornos con clústers

Consulte el capítulo "Using Other Databases" en el *Getting Started Guide* para mayor información.

El administrador de persistencia, la oficina postal y el administrador de usuarios JMS interactúan con el almacenamiento persistente. El administrador de persistenccia maneja la persistencia relacionada con los mensajes. La oficina postal maneja el enlace de la persistencia relacionada. El administrador de usuarios JMS maneja la persistencia relacionada con los usuarios. Todas las configuraciones para estos beans administrados se maneja en el archivo **`<your database type>-persistence-service.xml`**.

Encontrará ejemplos de archivos de configuración para las bases de datos MySQL, Oracle, PostgreSQL, Microsoft SQL Server y Sybase en el directorio **`jboss-as/docs/examples/jms`** del lanzamiento.

Para habilitar el soporte para una de estas bases de datos, reemplace el archivo de configuración predeterminado **`jboss-as/server/$PROFILE/deploy/messaging/hsqldb-persistence-service.xml`** con el archivo de configuración especifico para el tipo de su base de datos y re-inicie el servidor.

Por defecto, los servicios de mensajería que dependen de un almacenamiento de datos están referenciando **`java:/DefaultDS`** para la fuente de datos. Para implementar una fuente de datos con un nombre JNDI diferente entonces necesita actualizar todos los atributos **`DataSource`** en el archivo de configuración de persistencia. En la distribución encontrará configuraciones de fuentes de datos de ejemplo.

5.4. Configuración de la oficina postal

La oficina postal enruta los mensajes a su destino o destinos. Mantiene los mapeos entre direcciones a las cuales se puede enviar el mensaje y la cola final. Por ejemplo cuando al enviar un mensaje con una dirección que representa una cola JMS, la oficina postal enruta el mensaje a esa cola JMS. Al enviar un mensaje con una dirección que representa un tema JMS, la oficina postal enruta el mensaje a un grupo de colas - uno para cada subscripción JMS.

La oficina postal también maneja la persistencia para el mapeo de direcciones.

Las oficinas postales de JBoss Messaging tienen en cuenta los clústers. En un clúster enrutarán de manera automática y obtendrán mensajes entre nodos con el fin de proporcionar temas y colas JMS completamente distribuídas.

Configure la oficina postal en el archivo **`<database type>-persistence-service.xml`**. Por ejemplo:

```
<mbean code="org.jboss.messaging.core.jmx.MessagingPostOfficeService"
  name="jboss.messaging:service=PostOffice"
  xmbean-dd="xmdesc/MessagingPostOffice-xmbean.xml">

  <depends optional-attribute-name="ServerPeer">
    jboss.messaging:service=ServerPeer
  </depends>

  <depends>
    jboss.jca:service=DataSourceBinding,name=DefaultDS
  </depends>

  <depends optional-attribute-name="TransactionManager">
    jboss:service=TransactionManager
  </depends>

  <!-- The name of the post office -->

  <attribute name="PostOfficeName">JMS post office</attribute>

  <!-- The datasource used by the post office to access it's
       binding information -->

  <attribute name="DataSource">java:/DefaultDS</attribute>

  <!-- If true will attempt to create tables and indexes on
       every start-up -->

  <attribute name="CreateTablesOnStartup">true</attribute>

  <!-- If true then we will automatically detect and reject
       duplicate messages sent during failover -->

  <attribute name="DetectDuplicates">true</attribute>

  <!-- The size of the id cache to use when detecting duplicate
       messages -->

  <attribute name="IDCacheSize">500</attribute>

  <attribute name="SqlProperties">
    CREATE_POSTOFFICE_TABLE=CREATE TABLE JBM_POSTOFFICE
      (POSTOFFICE_NAME VARCHAR(255),
      NODE_ID INTEGER, QUEUE_NAME VARCHAR(255), COND VARCHAR(1023),
      SELECTOR VARCHAR(1023), CHANNEL_ID BIGINT, CLUSTERED CHAR(1),
      ALL_NODES CHAR(1),
      PRIMARY KEY(POSTOFFICE_NAME, NODE_ID, QUEUE_NAME)) ENGINE = INNODB

    INSERT_BINDING=INSERT INTO JBM_POSTOFFICE
      (POSTOFFICE_NAME, NODE_ID, QUEUE_NAME, COND, SELECTOR,
       CHANNEL_ID, CLUSTERED, ALL_NODES)
      VALUES (?, ?, ?, ?, ?, ?, ?, ?)

    DELETE_BINDING=DELETE FROM JBM_POSTOFFICE WHERE
      POSTOFFICE_NAME=? AND NODE_ID=? AND QUEUE_NAME=?

    LOAD_BINDINGS=SELECT QUEUE_NAME, COND, SELECTOR,
      CHANNEL_ID, CLUSTERED, ALL_NODES FROM
   JBM_POSTOFFICE WHERE POSTOFFICE_NAME=? AND NODE_ID=?
  </attribute>

  <!-- This post office is clustered. If you don't want a clustered post
       office then set to false -->

  <attribute name="Clustered">true</attribute>

  <!-- All the remaining properties only have to be specified if the post
       office is clustered. You can safely comment them out if your post
```

```
      office is non clustered -->

  <!-- The JGroups group name that the post office will use -->

  <attribute name="GroupName">
    ${jboss.messaging.groupname:MessagingPostOffice}
  </attribute>

  <!-- Max time to wait for state to arrive when the post office
       joins the cluster -->

  <attribute name="StateTimeout">5000</attribute>

  <!-- Max time to wait for a synchronous call to node members using
       the MessageDispatcher -->

  <attribute name="CastTimeout">50000</attribute>

  <!-- Set this to true if you want failover of connections to occur
       when a node is shut down -->

  <attribute name="FailoverOnNodeLeave">false</attribute>

  <!-- JGroups stack configuration for the data channel - used for sending
       data across the cluster -->

  <!-- By default we use the TCP stack for data -->
  <attribute name="DataChannelConfig">
    <config>
      <TCP start_port="7900"
        loopback="true"
        recv_buf_size="20000000"
        send_buf_size="640000"
        discard_incompatible_packets="true"
        max_bundle_size="64000"
        max_bundle_timeout="30"
        use_incoming_packet_handler="true"
        use_outgoing_packet_handler="false"
        down_thread="false" up_thread="false"
        enable_bundling="false"
        use_send_queues="false"
        sock_conn_timeout="300"
        skip_suspected_members="true"/>
      <MPING timeout="4000"
        bind_to_all_interfaces="true"
        mcast_addr="${jboss.messaging.datachanneludpaddress:228.6.6.6}"
        mcast_port="${jboss.messaging.datachanneludpport:45567}"
        ip_ttl="8"
        num_initial_members="2"
        num_ping_requests="1"/>
      <MERGE2 max_interval="100000"
        down_thread="false" up_thread="false" min_interval="20000"/>
      <FD_SOCK down_thread="false" up_thread="false"/>
      <VERIFY_SUSPECT timeout="1500" down_thread="false"
       up_thread="false"/>
      <pbcast.NAKACK max_xmit_size="60000"
        use_mcast_xmit="false" gc_lag="0"
        retransmit_timeout="300,600,1200,2400,4800"
        down_thread="false" up_thread="false"
        discard_delivered_msgs="true"/>
      <pbcast.STABLE stability_delay="1000" desired_avg_gossip="50000"
        down_thread="false" up_thread="false"
        max_bytes="400000"/>
      <pbcast.GMS print_local_addr="true" join_timeout="3000"
        down_thread="false" up_thread="false"
        join_retry_timeout="2000" shun="false"
        view_bundling="true"/>
    </config>
```

```
  </attribute>

  <!-- JGroups stack configuration to use for
       the control channel - used for control messages -->

  <!-- We use udp stack for the control channel -->
  <attribute name="ControlChannelConfig">
    <config>
      <UDP
        mcast_addr="${jboss.messaging.controlchanneludpaddress:228.7.7.7}"
        mcast_port="${jboss.messaging.controlchanneludpport:45568}"
        tos="8"
        ucast_recv_buf_size="20000000"
        ucast_send_buf_size="640000"
        mcast_recv_buf_size="25000000"
        mcast_send_buf_size="640000"
        loopback="false"
        discard_incompatible_packets="true"
        max_bundle_size="64000"
        max_bundle_timeout="30"
        use_incoming_packet_handler="true"
        use_outgoing_packet_handler="false"
        ip_ttl="2"
        down_thread="false" up_thread="false"
        enable_bundling="false"/>
      <PING timeout="2000"
        down_thread="false" up_thread="false" num_initial_members="3"/>
      <MERGE2 max_interval="100000"
        down_thread="false" up_thread="false" min_interval="20000"/>
      <FD_SOCK down_thread="false" up_thread="false"/>
      <FD timeout="10000" max_tries="5" down_thread="false"
        up_thread="false" shun="true"/>
      <VERIFY_SUSPECT timeout="1500" down_thread="false"
       up_thread="false"/>
      <pbcast.NAKACK max_xmit_size="60000"
        use_mcast_xmit="false" gc_lag="0"
        retransmit_timeout="300,600,1200,2400,4800"
        down_thread="false" up_thread="false"
        discard_delivered_msgs="true"/>
      <UNICAST timeout="300,600,1200,2400,3600"
        down_thread="false" up_thread="false"/>
      <pbcast.STABLE stability_delay="1000" desired_avg_gossip="50000"
        down_thread="false" up_thread="false"
        max_bytes="400000"/>
      <pbcast.GMS print_local_addr="true" join_timeout="3000"
        use_flush="true" flush_timeout="3000"
        down_thread="false" up_thread="false"
        join_retry_timeout="2000" shun="false"
        view_bundling="true"/>
      <FRAG2 frag_size="60000" down_thread="false" up_thread="false"/>
      <pbcast.STATE_TRANSFER down_thread="false" up_thread="false"
        use_flush="true" flush_timeout="3000"/>
      <pbcast.FLUSH down_thread="false" up_thread="false" timeout="20000"
        auto_flush_conf="false"/>
    </config>
  </attribute>

</mbean>
```

5.4.1. Atributos de la oficina postal

Los siguientes atributos están disponibles para configurar la oficina postal de mensajería:

DataSource

Especifica la fuente de datos para persistir los datos mapeados de la oficina postal.

SQLProperties

Especifica el DDL y el DML para una base de datos en particular. Si este atributo no se sobreescribe entonces se utilizará la configuración Hypersonic predeterminada.

Importante

Hypersonic no se soporta para entornos de producción. Este valor se debe cambiar para uso en producción.

CreateTablesOnStartup

Especifica si las tablas e índices se crean cuando se inicia el servicio de la oficina postal. Cuando se establece como **`true`**, la oficina postal creará las tablas e índices al iniciar. Si las tablas o los índices ya existen el controlador JDBC presentará una excepción **`SQLException`** y será ignorada por el administrador de persistencia permitiéndole continuar. El valor predeterminado es **`true`**.

DetectDuplicates

Especifica si la oficina postal detectará mensajes duplicados enviados cuando la entrega es restringida en un nuevo nodo después de un fallo en el servidor. Cuando se configura como **`true`**, la oficina postal detecta mensajes duplicados . El valor predeterminado es **`true`**.

IDCacheSize

Especifica el número de IDs de mensajes que el caché ID debe mantener. Si tiene lugar la conmutación de servidores en caso de fallo entonces el caché ID se accede como parte del proceso para prevenir que se envíen mensajes duplicados después de la conmutación de servidores. El valor predeterminado es **`500`** (mensajes).

PostOfficeName

Especifica el nombre de la oficina postal.

NodeIDView

Retorna los IDs de nodos de todos los nodos en el clúster como grupo.

GroupName

Especifica el nombre de la oficina postal al cual enlazar con otras oficinas postales con nombres idénticos.

Nota

Las oficinas postales con el mismo nombre de grupo formarán un clúster juntas. Asegúrese de que el nombre de grupo de la oficina postal coincida con todos los nodos con los que quiere formar un clúster.

Clustered

Especifica si la oficina postal se unirá a un clúster para formar temas y colas distribuídas. Si es **`false`** entonces todos los atributos relacionados con el clúster se ignorarán.

StateTimeout

Especifica el periodo máximo que la oficina postal esperará para recibir el estado del grupo cuando un nodo se une a un clúster que ya existe. El valor predeterminado es **`30000`**

milisegundos.

CastTimeout

Especifica el tiempo máximo que la oficina postal esperará por un mensaje de vaciado de respuesta de manera sincrónica. El valor predeterminado es **`30000`** milisegundos.

FailoverOnNodeLeave

Especifica la manera en que las conexiones se manejan cuando un nodo del servidor se apaga. Si se configura como **`true`** entonces las conexiones asociadas con el nodo del servidor que ha terminado de manera limpia conmutarán a otro nodo en caso de fallo. El valor predeterminado es **`false`**.

MaxConcurrentReplications

Especifica el número máximo de peticiones de replicación concurrentes a retornar antes de que las respuestas se bloqueen. Esto previene que JGroups se sobrecargue. El valor predeterminado es **`50`**.

Nota

No debe modificar el valor predeterminado para MaxConcurrentReplications. El valor predeterminado es la configuración óptima.

ControlChannelConfig

Especifica la configuración de la pila JGroups para el canal de control. El canal de control envía peticiones y recibe respuestas de otros nodos en el clúster.

Nota

JBoss Messaging utiliza JGroups para toda la administración del grupo. Se utiliza la configuración estándar de JGroups.

DataChannelConfig

Especifica la configuración de la pila JGroups para el canal de datos. El canal de datos envía y recibe mensajes de otros nodos en el clúster y replica datos de sesión.

Nota

JBoss Messaging utiliza JGroups para toda la administración del grupo. Se utiliza la configuración estándar de JGroups.

5.5. Configuración del administrador de persistencia

JBoss Messaging se envía junto con un administrador de persistencia JDBC utilizado para manejar la persistencia de datos de mensajes en una base de datos relacional accedida a través de JDBC. El administrador de persistencia es enchufable al servidor de mensajería y esto hace posible el proporcionar otras implementaciones para persisitir datos de mensajes en almacenamientos no relacionales, almacenamientos de archivos.

La configuración de servicios persistentes está agrupada en**`<database type>-persistence-`**

`service.xml`. Por defecto, JBoss Messaging se envía junto con el archivo **`hsqldb-persistence-service.xml`**, el cual configura el servidor Messaging a utilizar en la instancia de la base de datos Hypersonic que viene por defecto con cualquier instancia de JBoss Enterprise Application Server.

Aviso

Hypersonic no se soporta para uso en un entorno de producción.

JBoss Messaging también se envía junto con configuraciones del administrador de persistencia para MySQL, Oracle, PostgreSQL, Sybase y el servidor Microsoft SQL. Los archivos de configuración de ejemplo (tal como **`mysql-persistence-service.xml`** y **`ndb-persistence-service.xml`**) se encuentran disponibles en el directorio **`jboss-as/docs/examples/jms`** del grupo del lanzamiento.

El administrador de persistencia JDBC usa SQL estándar como su lenguaje de manipulación de datos (DML del inglés Data Manipulation Language), así que el escribir una configuración del administrador de persistencia para otro tipo de base de datos es solo cambiar el lenguaje de definición de datos (DDL del inglés Data Definition Language) de la configuración , lo cual usualmente es diferente para cada base de datos.

JBoss Messaging también se envía con una opción de configuración del asministrador de persistencia nulo, la cual se puede utilizar cuando no quiere nada de persistencia.

El archivo predeterminado de configuración del administrador de persistencia de Hypersonic es el siguiente:

```
<mbean code="org.jboss.messaging.core.jmx.JDBCPersistenceManagerService"
  name="jboss.messaging:service=PersistenceManager"
  xmbean-dd="xmdesc/JDBCPersistenceManager-xmbean.xml">

  <depends>jboss.jca:service=DataSourceBinding,name=DefaultDS</depends>

  <depends optional-attribute-name="TransactionManager">
    jboss:service=TransactionManager
  </depends>

  <!-- The datasource to use for the persistence manager -->

  <attribute name="DataSource">java:/DefaultDS</attribute>

  <!-- If true will attempt to create tables and indexes on every start-up -->

  <attribute name="CreateTablesOnStartup">true</attribute>

  <!-- If true then will use JDBC batch updates -->

  <attribute name="UsingBatchUpdates">false</attribute>

  <!-- The maximum number of parameters to include in a prepared statement -->

  <attribute name="MaxParams">500</attribute>
</mbean>
```

Importante

El tamaño máximo de los tipos de datos de imagen y texto de la base de datos Sybase se configura como **`2 kilobytes`** por defecto. Cualquier mensaje que exceda este límite es truncado sin ninguna información o advertencia. Configure el parámetro de la base de datos **`@@TEXTSIZE`** con un valor más alto para prevenir que potencialmente se trunquen los mensajes.
Este truncado también puede tener lugar en el servidor Microsoft SQL Server si el valor **`@@TEXTSIZE`** se establece como un valor menor que el valor predeterminado. Para mayor información, consulte http://jira.jboss.com/jira/browse/SOA-554.

Importante

El servidor Microsoft SQL no desasigna automáticamente el espacio de disco duro cuando se borran datos de una base de datos. Cuando se utiliza el espacio de la base de datos del disco duro como almacenamiento de datos para un servicio que almacena temporalmente muchos registros (tal como el servicio de mensajería), el espacio de disco se volverá mucho más grande más rápidamente que la cantidad de datos que de hecho se está almacenando.
Los administradores de la base de datos deben implementar planes de mantenimiento de la base de datos para asegurarse de reclamar el espacio que no esté siendo utilizado. Consulte la documentación del servidor Microsoft SQL para ver los comandos DBCC **`ShrinkDatabase`** y **`UpdateUsage`** para obtener información sobre cómo reclamar el espacio no utilizado. Para mayor información sobre este problema consulte https://jira.jboss.org/jira/browse/SOA-629

5.5.1. Atributos MBean PersistenceManager

Los siguientes atributos están disponibles para configurar el MBean administrador de persistencia:

CreateTablesOnStartup

Especifica si debe intentar la creación de tablas e índices cuando se inicia el administrador de persistencia. Cuando esta configurado como **`true`** (por defecto) el administrador de persistencia intentará crear tablas (e índices) cuando inicia. Si las tablas o índices ya existen el controlador JDBC presentará una **`SQLException`** y el administrador de persistencia lo ignorará, permitiéndole continuar.

UsingBatchUpdates

Especifica si múltiples actualizaciones de la base de datos se deben agrupar para mejorar el rendimiento. Configure este valor como **`true`** si su base de datos soporta actualizaciones en grupos JDBC. El valor predeterminado es **`false`**.

UsingBinaryStream

Especifica si los mensajes se almacenan y leen con un flujo binario JDBC en lugar de por medio de **`getBytes()`** y **`setBytes()`**. Configure este valor como **`false`** si su base de datos debe utilizar **`getBytes()`** y **`setBytes()`**. El valor predeterminado es **`true`**.

UsingTrailingByte

Especifica la manera en que se deben manejar los BLOBs de la base de datos Sybase que contienen ceros al final. Cuando se configura como **`true`** , se agrega un byte que no sea cero al final a cada BLOB antes de la persistencia y se borra del BLOB después de la persistencia, lo cual previene que la base de datos se trunque. El valor predeterminado es **`false`**

Nota

Ciertas versiones de Sybase truncan un BLOB con ceros. Este atributo solo se requiere si está ejecutando una base de datos Sybase.

SupportsBlobOnSelect

Especifica la manera en que los BLOBs se insertan en ciertos tipos de bases de datos. Cuando se establece como **`false`** entonces se utilizará la inserción en dos etapas. El valor predeterminado es **`true`**.

Nota

Ciertas bases de datos, especificamente Oracle, no permiten el insertar BLOBs por medio de una declaración **`INSERT INTO ... SELECT FROM`** y requieren una inserción con un mensaje condicional de dos etapas. Configure este atributo como **`false`** si está ejecutando una base de datos Oracle u otra base de datos con este requermiento.

SQLProperties

Especifica el DDL y el DML para una base de datos en particular. Si una declaración DDL o DML en particular no se sobreescribe, la configuración Hypersonic predeterminada se utilizará para esa declaración.

MaxParams

Especifica el número máximo de parámetros permitidos por declaración preparada al cargar mensajes. El valor predeterminado es **`500`**.

UseNDBFailoverStrategy

Especifica si una declaración SQL se debe volver a ejecutar en el evento de que falle al guardar los cambios de una transacción en una base de datos en un entorno con clústers. Si se configura como **`true`**, la declaración SQL se vuelve a ejecutar en caso de que falle al guardar los cambios. Si ocurren más errores, el administrador de persistencia asume que el error se debe a que la transacción anterior guardó los cambios de manera exitosa e ignora el error. Por defecto, este atributo se establece como **`false`**.

Nota

Cuando algunas bases de datos tal como MySQL, ejecutan en entornos en clústers, pueden fallar al guardar los cambios de transacciones de la base de datos. Si esto ocurre entonces el estado final de la transacción es desconocido.

5.6. Configuración del administrador de usuarios de JMS

El administrador de usuarios JMS mapea los IDs de clientes pre-configurados a los usuarios. También administra las tablas de usuarios y roles, dependiendo del módulo login que ha configurado.

Este es un ejemplo de la configuración del **`JMSUserManager`**

```
<mbean code="org.jboss.jms.server.plugin.JDBCJMSUserManagerService"
  name="jboss.messaging:service=JMSUserManager"
  xmbean-dd="xmdesc/JMSUserManager-xmbean.xml">
  <depends>jboss.jca:service=DataSourceBinding,name=DefaultDS</depends>
  <depends optional-attribute-name="TransactionManager">
    jboss:service=TransactionManager
  </depends>
  <attribute name="DataSource">java:/DefaultDS</attribute>
  <attribute name="CreateTablesOnStartup">true</attribute>
  <attribute name="SqlProperties">
    CREATE_USER_TABLE=CREATE TABLE JBM_USER (USER_ID VARCHAR(32) NOT NULL,
      PASSWD VARCHAR(32) NOT NULL, CLIENTID VARCHAR(128),
      PRIMARY KEY(USER_ID)) ENGINE = INNODB

    CREATE_ROLE_TABLE=CREATE TABLE JBM_ROLE (ROLE_ID VARCHAR(32) NOT NULL,
      USER_ID VARCHAR(32) NOT NULL, PRIMARY KEY(USER_ID, ROLE_ID))
      ENGINE = INNODB

    SELECT_PRECONF_CLIENTID=SELECT CLIENTID FROM JBM_USER WHERE USER_ID=?

    POPULATE.TABLES.1=INSERT INTO JBM_USER (USER_ID,PASSWD,CLIENTID)
      VALUES ('jdoe','jdoepw','jdoe-id')
  </attribute>
</mbean>
```

5.6.1. Atributos Bean administrados JMSUserManager

CreateTablesOnStartup

Especifica si se intenta crear las tablas e índices cuando inicia el MBean JMSUserManager. Cuando se configura como **`true`** (valor predeterminado), el JMSUserManager tratará de crear las tablas (e índices) en el arranque. Si las tablas o índices ya existen el controlador JDBC presentará una **`SQLException`** y el administrador de presistencia la ignorará, permitiéndole continuar sin problemas.

UsingBatchUpdates

Especifica si múltiples actualizaciones de la base de datos se agrupan en grupos para mejorar el rendimiento. Configure este valor como **`true`** si su base de datos soporta actualizaciones en grupo JDBC. El valor predeterminado es **`false`**.

SQLProperties

Especifica el DDL y el DML para una base de datos en particular. Si una declaración DDL o DML en particular no se sobreescribe, la configuración Hypersonic predeterminada se utilizará para esa declaración.

También puede insertar datos de usuarios predeterminados y de propiedades de roles adjuntandole el prefijo **`POPULATE.TABLES`** a los datos.

5.7. Configuración de destinos

5.7.1. Destinos pre-configurados

JBoss Messaging se envía junto con un grupo predeterminado de destinos pre-configurados que se implementarán durante el arranque del servidor. La información sobre la configuración de estos destinos se puede encontrar em la siguiente sección de **`destinations-service.xml`**:

```
<!--
      The Default Dead Letter Queue. This destination is a dependency of an EJB MDB
container.
   -->

   <mbean code="org.jboss.jms.server.destination.QueueService"
      name="jboss.messaging.destination:service=Queue,name=DLQ"
      xmbean-dd="xmdesc/Queue-xmbean.xml">
      <depends optional-attribute-
name="ServerPeer">jboss.messaging:service=ServerPeer</depends>
      <depends>jboss.messaging:service=PostOffice</depends>
   </mbean>

   <!--
      The Default Expiry Queue.
   -->

   <mbean code="org.jboss.jms.server.destination.QueueService"
      name="jboss.messaging.destination:service=Queue,name=ExpiryQueue"
      xmbean-dd="xmdesc/Queue-xmbean.xml">
      <depends optional-attribute-
name="ServerPeer">jboss.messaging:service=ServerPeer</depends>
      <depends>jboss.messaging:service=PostOffice</depends>
   </mbean>
```

5.7.2. Configuración de colas

5.7.2.1. Atributos MBean cola

Nombre

Define el nombre de la cola.

JNDIName

Define el nombre JNDI que enlaza la cola.

DLQ

Define el DLQ (del inglés Dead Letter Queue) utilizado para esta cola. Sobreescribe cualquier grupo de valores en el archivo de configuración del ServerPeer

ExpiryQueue

Define la cola de expiración y sobreescribe cualquier grupo de valores en el archivo de configuración del servidor compañero.

RedeliveryDelay

Define el retraso de re-entrega a utilizar para esta cola. Sobreescribe cualquier valor en el archivo de configuración del servidor compañero.

MaxDeliveryAttempts

Define el número máximo de veces que se intentará entregar un mensaje antes de enviar el mensaje al DLQ, si está configurado. El valor predeterminado, **-1**, significa que se utiliza el valor del archivo de configuración del servidor compañero. Cualquier otra configuración sobreescribe el valor establecido en el archivo de configuración del servidor compañero.

CreatedProgrammatically

Retorna **true** si la cola fue creada programáticamente.

MessageCount

Retorna el número total de mensajes en la cola. Es decir el número de mensajes que se están programando más el número de los que se están entregando más el número de los que no se entregan.

ScheduledMessageCount

Retorna el número de *mensajes programados* en la cola. Este es el número de mensajes programados para que sean entregados en una fecha posterior.

La entrega programada le permite especificar el momento más temprano en el que se entregará un mensaje en particular. Por ejemplo, puede enviar un mensaje ahora y especificar que no se entregue sino hasta dentro de dos horas. Para lograr esto, configure lo siguiente en el encabezado del mensaje:

```
long now = System.currentTimeMillis();

Message msg = sess.createMessage();

msg.setLongProperty(JBossMessage.JMS_JBOSS_SCHEDULED_DELIVERY_PROP_NAME,
  now + 1000 * 60 * 60 * 2);

prod.send(msg);
```

MaxSize

Se especifica el número máximo de mensajes que se pueden mantener en una cola. Cualquier mensaje que llegue después de este punto se borrará. Por defecto es **-1** que significa desvinculado.

Clustered

Este atributo se debe configurar como **true** si el destino es un clúster.

MessageCounter

Cada cola mantiene un contador de mensajes.

MessageCounterStatistics

Las estadísticas para el contador de mensajes

MessageCounterHistoryDayLimit

El número máximo de días que se mantiene el historial del contador de mensajes. Sobreescribe cualquier valor establecido en el archivo de configuración del servidor compañero.

ConsumerCount

El número de consumidores actualmente consumiendo de la cola.

DropOldMessageOnRedeploy

Especifica la manera en que se manejan los servicios de cola con los atributos del clúster que difieren de los atributos implementados anteriormente. Si se establece como **true**, todos los mensajes que quedan en la cola se borran después de la re-implementación del servicio de colas si el atributo del servicio de colas contiene un atributo clúster diferente. Si se configura como **false** (predeterminado), todos los mensajes se reservan.

Advertencia - Consideraciones al volver a implementar

Cuando vuelve a implementar un destino, debe apagar todos los nodos en el clúster, realizar los cambios apropiados a la configuración y luego re-iniciar los nodos.
El volver a implementar desde la cola sin clústers a una cola con clústers requiere que configure el atributo clúster como **`true`** y debe agregar la configuración de servicio de cola para cada nodo.
El volver a implementar desde una cola con clústers a una sin clústers requiere que configure el atributo clúster como **`false`** en una de las configuraciones de cola y debe borrar las otras colas en el clúster.

5.7.2.1.1. Configuración de seguridad del destino

<SecurityConfig> determina que roles se permite leer, escribir y crear en el destino. Tiene exactamente la misma sintaxis y semántica que la configuración de seguridad en destinos JBossMQ.

```
<SecurityConfig>
  <security>
    <role read="true" write="true" create="true"/>
  </security>
<SecurityConfig>
```

El elemento <SecurityConfig> debe contener un elemento <security>, el cual contiene múltiples elementos <role>. Cada elemento <role> define el tipo de accesso para ese rol en particular usando los siguientes atributos.

`read`
: Especifica que el rol puede crear clientes, recibir mensajes y navegar el destino.

`write`
: Especifica que el rol puede crear productores o enviar mensajes al destino.

`create`
: Especifica que el rol puede crear suscripciones durables en este destino.

Nota

La configuración de seguridad para un destino es opcional. Si no se especifica un elemento **`SecurityConfig`** entonces se utilizará la configuración predeterminada de seguridad del servidor compañero.

5.7.2.1.2. Parámetros de paginación de destinos

Pageable Channels es una funcionalidad de JBoss Messaging que le permite especificar el número máximo de mensajes que se pueden almacenar en la memoria en cualquier momento cola-por-cola, o tema-por-tema. Luego JBoss Messaging pagina los mensajes hacia y desde el almacenamiento de manera transparente en bloques, permitiéndoles crecer a las colas y a las subscripciones hasta tamaños muy grandes sin ninguna disminución en el rendimiento ya que el tamaño del canal se incrementa.

Los parámetros individuales asociados con los canales paginables son los siguientes:

`FullSize`
: Especifica el número máximo de mensajes que las subscripciones de tema o cola mantienen en la memoria en cualquier momento. La cola misma puede mantener muchos más mensajes

que esto, pero estos se paginan para el almacenamiento y desde él cuando sea necesario en el momento que se añadan o se consuman los mensajes. Si no se especifica, el valor predeterminado es **75000**.

PageSize

Especifica el número máximo de mensajes que se pre-cargan por operación al cargar los mensajes desde la cola o suscripción. Si no se especifica un valor entonces el predeterminado es **2000**.

DownCacheSize

Especifica el número máximo de mensajes que el Down Cache mantiene antes de que los mensajes se vacíen al almacenamiento. El valor predeterminado es **2000** mensajes.

Al paginar mensajes al almacenamiento desde la cola primero van a un *Down Cache* antes de escribirlos en el almacenamiento. Esto activa la escritura para que ocurra como una sola operación, lo cual ayuda al rendimiento.

Nota

Si quiere especificar los parámetros de paginación utilizados para colas temporales entonces necesita especificarlos en la fábrica de conexiones apropiada. Consulte Sección 5.8, “Configuración de fábricas de conexión ” para obtener más detalles sobre las diferentes fábricas de conexiones disponibles.

5.7.2.1.3. Operaciones Bean administradas en cola

RemoveAllMessages

Remueve (y borra) todos los mensajes de la cola.

Importante

Utilice esto con precaución ya que borrará permanentemente todos los mensajes de la cola.

ListAllMessages

Lista todos los mensajes actualmente en la cola. Utilizando un selector JMS como argumento en esta operación le permite recuperar un sub-grupo de mensajes en la cola que coinciden con los criterios dados.

ListDurableMessages

Lista todos los mensajes *durables* en la cola. Utilizando un selector JMS como argumento en esta operación le permite recuperar un subgrupo de mensajes en la cola que coinciden con los criterios dados.

ListNonDurableMessages

Lista todos los mensajes *no durables* en la cola. Utilizando un selector JMS como argumento le permite recuperar un subgrupo de mensajes en la cola que coinciden con los criterios dados.

ResetMessageCounter

Reestablece el contador de mensajes a cero.

ResetMessageCounterHistory

Reestablece el historial del contador de mensajes.

ListMessageCounterAsHTML

Lista el contador de mensajes en un formato HTML.

ListMessageCounterHistoryAsHTML

Lista el historial del contador de mensajes en un formato HTML.

5.7.3. Configuración de temas

5.7.3.1. Atributos Bean administrados por temas

Nombre

Define el nombre del tema

JNDIName

Define la ubicación JNDI en donde se enlaza el tema.

DLQ

Define el DLQ (del inglés Dead Letter Queue) utilizado para este tema. Sobreescribe cualquier valor en el archivo de configuración del servidor comapero.

ExpiryQueue

Define la cola de expiración utilizada para este tema. Sobreescribe cualquier valor establecido en el archivo de configuración del servidor compañero.

RedeliveryDelay

Define el periodo de retraso entre los intentos de reentrega para este tema. Sobreescribe cualquier grupo de valores en el archivo de configuración del servidor compañero.

MaxDeliveryAttempts

Define el número máximo de veces que se intentará entregar un mensaje antes de enviar el mensaje al DLQ, si está configurado. Elvalor predeterminado es **-1**, el cual especifica que el valor del archivo de configuración del servidor compañero que se va a utilizar. Cualquier otra configuración sobreescribe el valor del servidor compañero.

CreatedProgrammatically

Retorna **true** si el tema se creó programáticamente

MaxSize

Especifica un número máximo de mensajes que se pueden mantener en una subscripción de tema. Cualquier mensaje que llegue después de este punto se borrará del tema. Por defecto es **-1** lo cual significa que no hay restricción en el tamaño.

Clustered

Configure esto como **true** si su destino es un clúster.

MessageCounterHistoryDayLimit

Define el número máximo de días que se mantiene el historial contador de mensajes.

Sobreescribe cualquier valor establecido en el archivo de configuración del servidor compañero.

MessageCounters

Retorna una lista de los contadores de mensajes para las subscriciones del tema.

AllMessageCount

Retorna el número total de mensajes en todas las subscripciones que pertenecen a este tema.

DurableMessageCount

Retorna el número total de mensajes *durables* en todas las subscripciones que pertenecen a este tema.

NonDurableMessageCount

Retorna el número total de mensajes *no durables* en todas las subscripciones de este tema.

DropOldMessageOnRedeploy

Especifica la manera en que se manejan los servicios de cola con los atributos del clúster que difieren de los atributos implementados anteriormente. Si se establece como **`true`**, todos los mensajes que quedan en la cola se borran después de la re-implementación del servicio de colas si el atributo del servicio de colas contiene un atributo clúster diferente. Si se configura como **`false`** (predeterminado), todos los mensajes se reservan.

Advertencia - Consideraciones al volver a implementar

Cuando vuelve a implementar un destino, debe apagar todos los nodos en el clúster, realizar los cambios apropiados a la configuración y luego re-iniciar los nodos.
El volver a implementar desde la cola sin clústers a una cola con clústers requiere que configure el atributo clúster como **`true`** y debe agregar la configuración de servicio de cola para cada nodo.
El volver a implementar desde una cola con clústers a una sin clústers requiere que configure el atributo clúster como **`false`** en una de las configuraciones de cola y debe borrar las otras colas en el clúster.

AllSubscriptionsCount

Retorna un conteo de todas las subscripciones que pertenecen a este tema.

DurableSubscriptionsCount

Retorna el conteo de todas las subscripciones durables que pertenecen a este tema.

NonDurableSubscriptionsCount

Retorna un conteo de todas las subscripciones no durables que pertenecen a este tema.

5.7.3.1.1. Configuración de seguridad del destino

<SecurityConfig> determina que roles se permite leer, escribir y crear en el destino. Tiene exactamente la misma sintaxis y semántica que la configuración de seguridad en destinos JBossMQ.

El elemento <SecurityConfig> debe contener un elemento <security>, el cual contiene múltiples elementos <role>. Cada elemento <role> define el tipo de accesso para ese rol en particular usando los siguientes atributos.

read

Especifica que el rol puede crear clientes, recibir mensajes y navegar el destino.

write

Especifica que el rol puede crear productores o enviar mensajes al destino.

create

Especifica que el rol puede crear suscripciones durables en este destino.

Nota

La configuración de seguridad para un destino es opcional. Si no se especifica un elemento **`SecurityConfig`** entonces se utilizará la configuración predeterminada de seguridad del servidor compañero.

5.7.3.1.2. Parámetros de paginación de destinos

Previamente para que una aplicación soportara una cola o suscripción, era necesario almacenar la cola por completo en la memoria. Esto no siempre era posible para las colas o suscripciones muy largas.

Loa canales paginables es una nueva funcionalidad de JBoss Messaging que le permite especificar el número máximo de mensajes que se pueden almacenar en la memoria en cualquier momento cola-por-cola, o tema-por-tema. Luego JBoss Messaging pagina los mensajes hacia y desde el almacenamiento de manera transparente en bloques, permitiéndoles crecer a las colas y a las subscripciones hasta tamaños muy grandes sin ninguna disminución en el rendimiento ya que el tamaño del canal se incrementa. Esto se ha probado con colas por encima de diez millones de mensajes de 2 kilobytes en hardware muy básico y tiene el potencial de escalar a número de mensajes mucho más grandes.

Los parámetros individuales asociados con los canales paginables son los siguientes:

FullSize

Especifica el número máximo de mensajes que las subscripciones de tema o cola mantienen en la memoria en cualquier momento. La cola misma puede mantener muchos más mensajes que esto, pero estos se paginan para el almacenamiento y desde él cuando sea necesario en el momento que se añadan o se consuman los mensajes. Si no se especifica, el valor predeterminado es **`75000`**.

PageSize

Especifica el número máximo de mensajes que se pre-cargan por operación al cargar los mensajes desde la cola o suscripción. Si no se especifica un valor entonces el predeterminado es **`2000`**.

DownCacheSize

Especifica el número máximo de mensajes que el Down Cache mantiene antes de que los mensajes se vacíen al almacenamiento. El valor predeterminado es **`2000`** mensajes.

Al paginar mensajes al almacenamiento desde la cola primero van a un *Down Cache* antes de escribirlos en el almacenamiento. Esto activa la escritura para que ocurra como una sola operación, lo cual ayuda al rendimiento.

Nota

Los parámetros de paginación utilizados para colas temporales se deben especificar en la fábrica de conexiones apropiadas. Consulte la configuración de la fábrica de conexiones para obtener más detalles.

5.7.3.2. Operaciones Bean administradas por tema

RemoveAllMessages

Remueve (y borra) todos los mensajes de las subscripciones que pertenecen a este tema.

Importante

Utilice esto con precaución ya que borrará permanentemente todos los mensajes del tema.

ListAllMessages

Lista todos los mensajes que pertenecen a una suscripción especificada. Utilizando un selector JMS como argumento en esta operación le permite recuperar un subgrupo de mensajes en la cola que coincidan con los criterios dados.

ListDurableMessages

Lista todos los mensajes durables que pertenecen a la suscripción especificada. Utilizando un selector JMS como argumento en esta operación le permite recuperar un subgrupo de mensajes en la cola que coinciden con los criterios dados.

ResetMessageCounter

Reestablece el contador de mensajes a cero.

ResetMessageCounterHistory

Reestablece el historial del contador de mensajes.

ListAllSubscriptionsAsHTML

Lista todas las subscripciones que pertenecen a este tema en un formato HTML.

ListDurableSubscriptionsAsHTML

Lista todas las subscripciones durables que pertenecen a este tema en un formato HTML.

ListNonDurableSubscriptions

Lista todos los mensajes no durables que pertenecen a la suscripción especificada Utilizando un selector JMS como argumento en esta operación le permite recuperar un subgrupo de mensajes en la cola que coinciden con los criterios dados.

ListNonDurableSubscriptionsAsHTML

Lista todas las subscripciones no durables de este tema en un formato HTML.

5.8. Configuración de fábricas de conexión

Con la configuración predeterminada JBoss Messaging enlaza dos fábricas de conexiones en JNDI durante el arranque.

La primera fábrica de conexiones es la fábrica de conexiones sin clústers predeterminada. Esta fárbica de conexiones se proporciona para mantener la compatibilidad con las aplicaciones que originalmente se escribieron frente a JBossMQ, el cual no tiene conmutación automática de servidor en caso de fallos ni balanceo de carga. Esta fábrica de conexiones se debe utilizar si no requiere conmutación de servidor en caso de fallos o balaceo de carga automáticos.

La primera fábrica de conexiones se enlaza a los siguientes contextos JNDI:

- **`/ConnectionFactory`**
- **`/XAConnectionFactory`**
- **`java:/ConnectionFactory`**
- **`java:/XAConnectionFactory`**.

La segunda fábrica de conexiones es la predeterminada en clústers y está enlazada a los siguientes contextos JNDI:

- **`/ClusteredConnectionFactory`**
- **`/ClusteredXAConnectionFactory`**
- **`java:/ClusteredConnectionFactory`**
- **`java:/ClusteredXAConnectionFactory`**

Si quiere brindar un ID de cliente predeterminado para una fábrica de conexiones o si quiere enlazarla a diferentes locationConsider JNDI entonces configure e implemente fábricas de conexiones diferentes. Para implementar una nueva fábrica de conexiones, configure un nuevo bean administrado **`ConnectionFactory`** en **`connection-factories-service.xml`**.

También es posible crear un nuevo descriptor de despliegue de servicios **`<`*`name`*`>-service.xml`** y desplegarla en **`$JBOSS_HOME/server/messaging/deploy`**.

Habilita el soporte para la conmutación de servidores automática en caso de fallo o el balanceo de carga configurando los atributos relevantes en su fábrica de conexiones:

Ejemplo 5.1. Fábrica de conexiones

Esta fábrica de conexiones de ejemplo crea una fábrica de conexión con el ID del cliente preconfigurado **`myClientID`**, el cual está enlazado a dos lugares en el árbol JNDI: **`/MyConnectionFactory`** y **`/factories/cf`**.

El ejemplo sobreescribe los siguientes valores predeterminados:

- **`PreFetchSize`**
- **`DefaultTempQueueFullSize`**
- **`DefaultTempQueuePageSize`**
- **`DefaultTempQueueDownCacheSize`**
- **`DupsOKBatchSize`**
- **`SupportsFailover`**
- **`SupportsLoadBalancing`**
- **`LoadBalancingFactory`**

La fábrica de conexiones usa el conector remoto predeterminado. Para utilizar un conector remoto diferente con la fábrica de conexiones, cambiel el atributo **`Connector`** para especificar el nombre del servicio del conector que desea utilizar.

```
<mbean code="org.jboss.jms.server.connectionfactory.ConnectionFactory"
  name="jboss.messaging.connectionfactory:service=MyConnectionFactory"
  xmbean-dd="xmdesc/ConnectionFactory-xmbean.xml">
  <depends optional-attribute-name="ServerPeer">
    jboss.messaging:service=ServerPeer
  </depends>
  <depends optional-attribute-name="Connector">
    jboss.messaging:service=Connector,transport=bisocket
  </depends>
  <depends>jboss.messaging:service=PostOffice</depends>

  <attribute name="JNDIBindings">
    <bindings>
    <binding>/MyConnectionFactory</binding>
    <binding>/factories/cf</binding>
    </bindings>
  </attribute>

  <attribute name="ClientID">myClientID</attribute>

  <attribute name="SupportsFailover">true</attribute>

  <attribute name="SupportsLoadBalancing">false</attribute>

  <attribute name="LoadBalancingFactory">
    org.acme.MyLoadBalancingFactory
  </attribute>

  <attribute name="PrefetchSize">1000</attribute>

  <attribute name="SlowConsumers">false</attribute>

  <attribute name="StrictTck">true</attribute>

  <attribute name="SendAcksAsync">false</attribute>

  <attribute name="DefaultTempQueueFullSize">50000</attribute>

  <attribute name="DefaultTempQueuePageSize">1000</attribute>

  <attribute name="DefaultTempQueueDownCacheSize">1000</attribute>

  <attribute name="DupsOKBatchSize">10000</attribute>
</mbean>
```

5.8.1. Atributos del Bean administrado ConnectionFactory

ClientID

Puede pre-configurar una fábrica de conexiones con un ID de cliente. Cualquier conexión creada utilizando esta fábrica de conexiones obtendrá este ID de cliente

JNDIBindings

Lista de enlaces disponibles JNDI para esta fábrica de conexiones.

PrefetchSize

Especifica cuántos mensajes puede tener una ventana en un momento dado para el control de flujo del consumidor. El tamaño de la ventana determina el número de mensajes que un servidor puede enviar a un consumidor sin bloquearse. Cada consumidor mantiene un buffer de mensajes desde donde consume.

El protocolo de control de transmisión TCP (del inglés Transmission Control Protocol) implementa su propio control de flujo adicional. El consumo de mensajes también se puede

bloquear si el tamaño de la ventana TCP es menor que el parámetro ***PrefetchSize***.

SlowConsumers

Especifica si el tamaño de buffer permitido para los consumidores lentos se reduce. El reducir el tamaño del buffer para los consumidores lentos minimiza el incremento del potencial para que los mensajes sean consumidos por consumidores rápidos. No es posible el inhabilitar por completo el buffer, pero el establecer el atributo **SlowConsumers** como **true** reducirá el tamaño del buffer. El configurar este atributo como **true** es equivalente a establecer **PrefetchSize** como **1**, el cual es el valor menor posible disponible.

StrictTck

Habilita el conmportamiento JMS estricto si el atributo se configura como **true**. El comportamiento JMS estricto lo requiere el kit de compatibilidad de tecnología (TCK del inglés Technology Compatibility Kit).

SendAcksAsync

Especifica que los reconocimientos se envían de manera asincrónica si el atributo se establece como **true**. Esto puede mejorar el rendimiento particularmente si el modo **auto_acknowledge** está activo.

DefaultTempQueueFullSize

Atributo opcional que especifica los parámetros de paginación para los destinos de cola temporales de tamaño completo, los cuales van para las conexiones creadas con esta fábrica de conexiones. El valor predeterminado es **200000**. Para obtener mayor información sobre estos atributos consulte Sección 5.7.3.1.2, “Parámetros de paginación de destinos ”.

DefaultTempQueuePageSize

Atributo opcional que especifica los parámetros de paginación para los destinos de tamaño de página temporal , los cuales van para las conexiones creadas con esta fábrica de conexiones. El valor predeterminado es **2000**. Para mayor información sobre estos atributos consulte Sección 5.7.3.1.2, “Parámetros de paginación de destinos ”.

DefaultTempQueueDownCacheSize

Atributo opcional que especifica los parámetros de paginación para los destinos de tamaño caché temporal, los cuales van para las conexiones creadas con esta fábrica de conexiones. El valor predeterminado es **2000**. Para obtener mayor información sobre estos atributos, consulte Sección 5.7.3.1.2, “Parámetros de paginación de destinos ”.

DupsOKBatchSize

Especifica el número de reconocimientos **DUPS_OK_ACKNOWLEDGE** que se mandan al buffer localmente antes de enviarlos. El valor predeterminado es **2000**

SupportsLoadBalancing

Especifica si el balanceo de carga del lado del cliente se encuentra activado en una fábrica de conexiones. Si si está habilitado entonces cualquier conexión creada con esa fábrica de conexiones tendrá balanceo de carga a través de los nodos del clúster. Una vez se crea una conexión en un nodo en particular, esta permanece en ese nodo. El valor predeterminado es **false**.

SupportsFailover

Especifica si la conmutación automática de servidores en caso de fallos del lado del cliente está habilitada para la fábrica de conexiones en instalaciones con clústers. Si la conmutación automática está habilitada entonces JBoss Messaging de manera automática y transparente

conmutará en caso de fallo a otro nodo en el clúster cuando se detecte un problema en la conexión. El valor predeterminado es **`false`**.

Nota

Cuando la conmutación automática en caso de fallo está inhabilitada el código del usuario es el responsable de atrapar excepciones de conexiones en operaciones JMS sincrónicas y debe instalar un **`ExceptionListener`** JMS para atrapar excepciones de manera asincrónica. Cuando se atrapa una conexión, el código del lado del cliente debe buscar una nueva fábrica de conexiones utilizando HAJNDI y recrea la conexión.

DisableRemotingChecks

Especifica si la fábrica de conexiones chequea que el JBoss Remoting Connector correspondiente tenga valores sensatos. JBoss Messaging es muy sensible a estos valores y casi nunca hay necesidad de cambiarlos. Para desactivar tal chequeo de sanidad configure **`DisableRemotingChecks`** como **`false`**. El valor predeterminado es **`true`**.\n \n

Aviso

No inhabilite los chequeos remotos; inestabilidad del sistema.

LoadBalancingFactory

Especifica la implementación de la fábrica de balanceo de carga del lado del cliente que la fábrica de conexiones utiliza. El valor debe corresponder al nombre de una clase que implementa la interfaz **`org.jboss.jms.client.plugin.LoadBalancingFactory`**.

El valor predeterminado es **`org.jboss.jms.client.plugin.RoundRobinLoadBalancingFactory`**, el cual balancea la carga de conexiones a través del clúster de una manera equitativa.

Conector

Especifica el conector remoto que la fábrica de conexiones utiliza. Diferentes fábricas de conexiones pueden utilizar diferentes conectores así que puede desplegar una fábrica de conexiones que use el transporte HTTP para comunicarse con el servidor y otra que utilice el transporte bisocket para comunicar.

EnableOrderingGroup

Especifica si el ordenamiento estricto de mensajes está habilitado en la **`ConnectionFactory`**. If set to **`true`**, cualquier mensajes enviado desde los productores, los cuales sean creados desde la fábrica de conexiones habilitada se vuelven mensajes de grupo ordenados. El valor predeterminado para este parámetro es **`false`**.

DefaultOrderingGroupName

Especifica el nombre predeterminado para el grupo de ordenamiento de mensajes. El nombre especificado tendrá efecto una vez que el parámetro **`EnableOrderingGroup`** se establezca como **`true`** . Si este atributo falta entonces el nombre del grupo se generará automáticamente.

5.9. Configuración del conector remoto

JBoss Messaging usa JBoss Remoting para toda la comunicación entre el cliente y el servidor.

Para mayor información sobre la configuración de JBoss Remoting y sus funcionalidades consulte el capítulo [Remoting] en el *JBoss Administration and Configuration Guide*.

La configuración predeterminada incluye un sólo conector remoto, el cual lo utiliza la única fábrica de conexiones predeterminada. Cada fábrica de conexiones se puede configurar para que utilice un conector diferente.

El conector predeterminado se configura para que utilice el transporte bisocket remoto. El transporte bisocket es un transporte con base en un socket TCP, el cual escucha y acepta conexiones sólo del lado del servidor. Es decir, las conexiones siempre se inician del lado del cliente. Esto significa que es ideal en un escenario de cortafuergos típico en donde se permiten las conexiones entrantes en el servidor o en donde sólo se permiten las conexiones salientes del cliente.

El transporte bisocket se puede configurar para utilizar SSL en donde se requiera un mayor nivel de seguridad.

El otro transporte soportado es el transporte HTTP. Este utiliza el protocolo de transferencia Hypertext para la comunicación entre cliente y servidor. El cliente sondea periódicamente el servidor en busca de mensajes para recibir datos. Este transporte es ideal para situaciones en donde hay un cortafuegos entre el cliente y el servidor, lo cual sólo permite tráfico HTTP de entrada en el servidor. Este transporte no tiene tanto rendimiento como el transporte bisocket debido a la naturaleza del sondeo y a las limitaciones de HTTP. No está diseñado para situaciones con bastante carga.

JBoss Messaging no soporta actualmente ningún otro transporte remoto.

Los detalles de configuración de Remoting se encuentran en **`$JBOSS_HOME/server/$SERVER/deploy/messaging/remoting-bisocket-service.xml`**. El siguiente código es un ejemplo de una configuración remota bisocket:

Ejemplo 5.2. Configuración remota bisocket

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
     Standard bisocket-based Remoting service deployment descriptor.

     $Id: remoting-bisocket-service.xml 3981 2008-03-28 18:00:41Z timfox $
 -->

<server>

   <!-- Standard bisocket connector - the bisocket transport only opens
connection from client->server
        so can be used with firewalls where only outgoing connections are
allowed.
        For examples of HTTP and SSL transports see docs/examples -->
   <mbean code="org.jboss.remoting.transport.Connector"
          name="jboss.messaging:service=Connector,transport=bisocket"
          display-name="Bisocket Transport Connector">
      <attribute name="Configuration">
         <config>
            <invoker transport="bisocket">

               <!-- There should be no reason to change these parameters -
warning!
                    Changing them may stop JBoss Messaging working correctly --
>
               <attribute name="marshaller"
isParam="true">org.jboss.jms.wireformat.JMSWireFormat</attribute>
               <attribute name="unmarshaller"
isParam="true">org.jboss.jms.wireformat.JMSWireFormat</attribute>
               <attribute name="dataType" isParam="true">jms</attribute>
               <attribute name="socket.check_connection"
isParam="true">false</attribute>
               <attribute
name="serverBindAddress">${jboss.bind.address}</attribute>
               <attribute
name="serverBindPort">${jboss.messaging.connector.bisocket.port:4457}</attribute>
               <attribute name="clientSocketClass"
isParam="true">org.jboss.jms.client.remoting.ClientSocketWrapper</attribute>
               <attribute
name="serverSocketClass">org.jboss.jms.server.remoting.ServerSocketWrapper</attr
ibute>
               <attribute
name="onewayThreadPool">org.jboss.jms.server.remoting.DirectThreadPool</attribut
e>

               <!-- the following parameters are useful when there is a
firewall between client and server. Uncomment them if so.-->
               <!--
               <attribute name="numberOfCallRetries"
isParam="true">1</attribute>
               <attribute name="pingFrequency"
isParam="true">214748364</attribute>
               <attribute name="pingWindowFactor" isParam="true">10</attribute>
               <attribute name="generalizeSocketException"
isParam="true">true</attribute>
               -->

               <!-- Now remoting supports socket write timeout configuration.
Uncomment this if you need it. -->
               <!--
               <attribute name="writeTimeout" isParam="true">30000</attribute>
               -->

               <!-- End immutable parameters -->

               <attribute name="stopLeaseOnFailure"
isParam="true">true</attribute>
```

```
               <!-- Periodicity of client pings. Server window by default is
twice this figure -->
               <attribute name="clientLeasePeriod"
isParam="true">10000</attribute>
               <attribute name="validatorPingPeriod"
isParam="true">10000</attribute>
               <attribute name="validatorPingTimeout"
isParam="true">5000</attribute>

               <attribute name="failureDisconnectTimeout"
isParam="true">0</attribute>
               <attribute name="callbackErrorsAllowed">1</attribute>
               <attribute name="registerCallbackListener">false</attribute>
               <attribute name="useClientConnectionIdentity"
isParam="true">true</attribute>

             <attribute name="timeout" isParam="true">0</attribute>

               <!-- Max Number of connections in client pool. This should be
significantly higher than
                    the max number of sessions/consumers you expect -->
               <attribute name="JBM_clientMaxPoolSize"
isParam="true">200</attribute>

               <!-- The maximum time to wait before timing out on trying to
write a message to socket for delivery -->
               <attribute name="callbackTimeout">10000</attribute>

               <!-- Use these parameters to specify values for binding and
connecting control connections to
                    work with your firewall/NAT configuration
               <attribute name="secondaryBindPort">xyz</attribute>
               <attribute name="secondaryConnectPort">abc</attribute>
               -->

            </invoker>
            <handlers>
               <handler
subsystem="JMS">org.jboss.jms.server.remoting.JMSServerInvocationHandler</handle
r>
            </handlers>
         </config>
      </attribute>
   </mbean>

</server>
```

Hay atributos restringidos que no se deben cambiar a menos de que esté completamente seguro de comprender el impacto de los cambios. Los siguientes atributos se pueden cambiar de manera segura para configurar los requerimientos de su proyecto:

clientLeasePeriod

Los clientes retornan periódicamente *heartbeats* al servidor para confirmarle al servidor que aún se encuentran activos. Si el servidor no recibe un heartbeat después de cierto tiempo entonces cerrará la conexión y removerá todos los recursos en el servidor correspondiente a la sesión del cliente. El **`clientLeasePeriod`** determina el periodo de tiempo entre heartbeats en milisegundos. El valor predeterminado es **`10000`**.

Por defecto, el servidor cierra un cliente si no recibe un heartbeat dentro del doble del **`clientLeasePeriod`**. En realidad, se ajusta el tamaño del periodo de acuerdo con la carga del sistema.

numberOfRetries

El número de segundos que JBoss Remoting bloqueará en el pool de clientes en espera de una conexión libre. Si tiene un número grande de sesiones accediendo a mismo tiempo al servidor desde un cliente y no puede obtener conexiones desde el pool, debería considerar el incrementar este valor.

clientMaxPoolSize

JBoss Remoting mantiene un pool del lado del cliente de conexiones TCP en las cuales prestar servicios a las peticiones. Si tiene un número grande de sesiones simultáneas accediendo el servidor desde un cliente y no puede obtener conexiones desde el pool, considere el incrementar este valor.

secondaryBindPort

El transporte bisocket utiliza conexiones de control para pasar mensajes de control entre el servidor y el cliente. Este atributo define la dirección a la que se enlaza el **`ServerSocket`** secundario. Para trabajar por detrás de un corta fuegos es posible que necesite especificar un valor en particular para su configuración del corta fuegos.

secondaryConnectPort

El puerto que el cliente utiliza para conectarse. Puede especificar esto para permitirle a los clientes trabajar con enrutadores NAT.

maxPoolSize

El número de hilos utilizados en el lado del servidor para atender peticiones.

Por defecto JBoss Messaging se enlaza a **`${jboss.bind.address}`**, el cual se puede definir ejecutando el comando **`./run.sh -c [yourconfig] -b [yourIP]`**.

Si es necesario, puede cambiar **`remoting-bisocket-service.xml`** parautilizar un puerto de comunicación diferente.

Aviso

No cambie los valores en la configuración del conector aparte de los listados anteriormente El cambiarlos puede causar que JBoss Messaging pare de funcionar correctamente.

5.10. ServiceBindingManager

El **`SeviceBindingManager`** proporciona múltiples instancias del servidor de aplicaciones ejecutando en la misma IP usando diferentes rangos de puerto, lo cual es útil durante el desarrollo. Hay otras maneras de lograr esto pero el **`ServiceBindingManager`** hace la mayor parte del trabajo duro.

Capítulo 6. Notas sobre clústers

Para ayudar a ubicar la información relacionada con los clústers, en esta parte del manual proporcionamos un resumen de cada consideración con enlaces a los componentes relacionados de JBoss Messaging.

6.1. Id único del servidor compañero

En la mayoría de los casos, JBoss Messaging funciona en un entorno con clústers con cambios mínimos en la configuración. Un cambio crucial que se debe realizar es el asignar un id único de servidor a cada nodo.

Todo nodo desplegado debe tener un id único incluyendo aquellos nodos que forman un clúster LAN y los nodos enlazados por puentes de mensajes.

El atributo ***ServerPeerID*** se utiliza para establecer esta información. Consulte Sección 5.2, “Atributos del ServerPeer ” para obtener mayor información.

6.2. Destinos con clústers

JBoss Messaging pone en clústers colas y temas JMS (del inglés Java Message Service) de manera transparente a través del clúster. Los mensajes enviados a un tema o a una cola distribuída en un nodo son consumibles en otros nodos. Para hacer un destino en particular en clúster se utiliza el atributo **`clustered`** para establecer esta funcionalidad. Consulte Sección 5.4.1, “Atributos de la oficina postal” para mayor información.

JBoss Messaging balancea los mensajes entre nodos satisfaciendo consumidores con diferentes velocidades de manera que el procesamiento de cargas se pueda distribuir eficiente a través del clúster.

Para inhabilitar la redistribución de mensajes entre nodos pero si aún quiere retener las otras características de los destinos en clústers no especifique el atributo **`ClusterPullConnectionFactoryName`** en el servidor compañero. Consulte Sección 5.2, “Atributos del ServerPeer ” para obtener mayores detalles sobre este atributo.

6.3. Subscripciones durables en clústers

Las subscripciones durables de JBoss Messaging también se pueden poner en clústers de manera que le permite a múltiples subscriptores en múltiplos nodos consumir de una subscripción durable. Una subscrición durable se pone en clúster automáticamente dado que su tema se encuentra en un clúster.

Para mayor información sobre la configurición de temas y colas en clústers, consulte el atributo ***Clustered*** en Sección 5.4.1, “Atributos de la oficina postal”

6.4. Destinos temporales en clústers

JBoss Messaging soporta temas y colas temporales en clústers. Todos los temas y colas temporales se pondrán en clústers si la oficina postal está en clúster.

Para obtener mayor información sobre la configuración de temas y colas en clústers consulte el atributo ***Clustered*** en Sección 5.4.1, “Atributos de la oficina postal”.

6.5. Servidores sin clústers

Establezca el atributo clúster ***PostOffice*** como **`false`** si no quiere que todos los nodos participen en un clúster o si no quiere que el servidor esté en clúster.

Para mayor información sobre la configuración de un servidor sin clúster, consulte los diferentes atributos en Sección 5.4.1, “Atributos de la oficina postal”.

6.6. Ordenamiento de mensajes en el clúster

Para asegurarse de que los mensajes se consumen en el mismo orden en que se produjeron, establezca un ordenamiento JMS estricto configurando el atributo del servidor compañero **`DefaultPreserveOrdering`** como **`true`**. Mientras esté configurado como **`true`**, los mensajes no se pueden distribuir tan libremente alrededor del clúster. El valor predeterminado es **`false`**.

6.7. Operaciones Idempotent

Se garantiza que un mensaje sea persistido cuando el mensaje enviado a un destino persistente retorna sin excepción.

Una excepción no garantiza que el mensaje *no* fue persistido ya que la falla puede haber ocurrido entre el momento en que el mensaje se estaba persistiendo y el momento en que una respuesta fuera enviada al que realiza la llamada.

Por lo tanto las aplicaciones deben estas codificadas de manera que las operaciones son *idempotent* — es decir, las operaciones se pueden repetir sin hacer que el sistema se vuelva inconsistente.

Puede implementar este comportamiento en el nivel de la aplicación buscando mensajes duplicados y descartádolos si el mensaje original se ha enviado de manera exitosa. Este *chequeo de duplicados* es una técnica poderosa que puede eliminar la necesidad de transacciones XA.

JBoss Messaging está configurado por defecto para realizar un chequeo de duplicados en un entorno con clústers.

Las consideraciones relacionadas con la persistencia se encuentran en Sección 5.2.1, “Métodos del ServerPeer ”, Sección 5.3, “Cambio de la base de datos ”, Sección 5.5, “Configuración del administrador de persistencia ” y Sección 8.1, “Sinopsis del puente de mensajes ”.

6.8. Fábricas de conexiones en clústers

Cuando **`supportsLoadBalancing`** está configurado como **`true`** en la fábrica de conexiones entonces los intentos consecutivos de creación de conexiones se resolverán de manera equitativa entre los servidores disponibles. El primer nodo se escoje al azar.

Cuando **`supportsFailover`** está configurado como **`true`**, la conmutación de servidores en caso de fallo ocurrirá de manera transparente y automáticamente cuando se detecte cualquier error de conexión.

Para mayor información sobre la configuración de fábricas de conexiones consulte Sección 5.8.1, “Atributos del Bean administrado ConnectionFactory”.

Capítulo 7. Configuración de JBoss Messaging XA Recovery

Esta sección describe la manera de configurar JBoss Transactions en JBoss Enterprise Application Platform para manejar la recuperación XA para recursos de JBoss Messaging.

El administrador de recuperación de transacciones de JBoss se puede configurar para sondear continuamente y poder recuperar recusos de JBoss Messaging XA. Esto proporciona un alto nivel de durabilidad de transacciones.

Para habilitar el administrador de recuperación de transacciones de JBoss, agréguele una línea a **`$JBOSS_HOME/server/$PROFILE/conf/jbossts-properties.xml`**. El siguiente pedazo de código incluye la línea requerida:

```
<properties depends="arjuna" name="jta">
  <!--
  Support subtransactions in the JTA layer?
  Default is NO.
  -->
  <property name="com.arjuna.ats.jta.supportSubtransactions" value="NO"/>
  <property name="com.arjuna.ats.jta.jtaTMImplementation"

value="com.arjuna.ats.internal.jta.transaction.arjunacore.TransactionManagerImple"/>
  <property name="com.arjuna.ats.jta.jtaUTImplementation"

value="com.arjuna.ats.internal.jta.transaction.arjunacore.UserTransactionImple"/>
  <!--
      *** Add this line to enable recovery for JMS resources using
DefaultJMSProvider ***
  -->
  <property name="com.arjuna.ats.jta.recovery.XAResourceRecovery.JBMESSAGING1"

value="org.jboss.jms.server.recovery.MessagingXAResourceRecovery;java:/DefaultJMSPr
ovider"/>

</properties>
```

Aquí el administrador de recuperación trata de recuperar recursos JMS utilizando el JMS Provider Loader, **`DefaultJMSProvider`**.

`DefaultJMSProvider` se envía junto con JBoss Enterprise Application Platform. Se define en **`$JBOSS_HOME/server/$PROFILE/conf/jms-ds.xml`** (or, in a clustered environment, **`hajndi-jms-ds.xml`**). Para realizar la recuperación con un cargador proveedor JMS diferente (por ejemplo, uno que corresponda con un proveedor JMS remoto), agregue otra línea al archivo de propiedades y especifique su proveedor remoto en lugar de **`DefaultJMSProvider`**. el nombre de su proveedor se debe listar en su archivo de configuración bean administrado.

Cada proveedor requiere un nombre único, por ejemplo, **`com.arjuna.ats.jta.recovery.XAResourceRecovery.JBMESSAGING1`**, **`com.arjuna.ats.jta.recovery.XAResourceRecovery.JBMESSAGING2`**, etc.

La recuperación debe funcionar con cualquier proveedor JMS que implemente XAResources recuperables (por ejemplo, implementa apropiadamente **`XAResource.recover()`**).

Para que el administrador de recuperación recupere desde cualquier nodo del clúster es necesario agregar una línea en l**`hajndi-jms-ds.xml`** para todos los nodos del clúster

Capítulo 8. Configuración del puente de mensajes de JBoss Messaging

8.1. Sinopsis del puente de mensajes

JBoss Messaging incluye un puente de mensajes completamente funcional

El puente consume mensajes de una cola fuente o tema y los envía a una cola destino o tema usualmente en un servidor diferente. Los servidores fuente y de destino no se tienen que encontrar en el mismo clúster, lo cual hace que el realizar el puente sea apropiado para enviar mensajes desde un clúster al otro (por ejemplo, a través de un WAN) y en donde puede que la conexión sea poco confiable.

Un puente se despliega como un bean administrado dentro de cualquier instancia de JBoss Enterprise Application Platform. Para implementar agregue el descriptor bean administrado en el directorio **`deploy`** de una configuración de la plataforma de aplicaciones empresariales que contenga JBoss Messaging.

El ejemplo en **`$JBOSS_HOME/docs/examples/jboss-messaging-examples/bridge/`** demuestra el despliegue de un puente simple en JBoss Enterprise Application Platform y el movimiento de mensajes del destino al destino.

El puente también se puede utilizar para recuperar mensajes desde servidores JMS que no son JBoss Messaging en tanto cumplan con los requeriminetos de JMS 1.1.

El puente tiene una capacidad de recuperación de fallos incorporada así que si la conexión al servidor fuente o al de destino se pierde, por ejemplo, debido a fallas en la red, el puente volverá a intentar conectarse a la fuente o al destino hasta que se reestablezca la red. Cuando se vuelve a conectar reiniciará su operación de manera normal.

El puente se puede configurar para consumir mensajes que coincidan un selector JMS en particular.

Se puede configurar para que consuma de una cola o tema. Cuando el puente consume de un tema se puede configurar para que consuma con una subscripción durable o no durable.

El puente se puede configurar para que maneje mensajes con uno de tres niveles de *calidad de servicio*, estos son:

Niveles de calidad de servicio del puente

`QOS_AT_MOST_ONCE`

Este modo especifica que los mensajes alcanzarán su destino máximo una vez. Los mensajes se consumen desde la fuente y se reconocen antes de enviarlos al destino. Por lo tanto existe la posibilidad de que se pierdan si ocurre una falla desde cuando deja la fuente hasta cuando llegan al destino. Por lo tanto la entrega ocurrirá máximo una vez.

Este modo está disponible para mensajes persistentes y no-persistentes

`QOS_DUPLICATES_OK`

Este modo especifica que los mensajes se consumen desde la fuente y luego se reconocen después de que se han enviado exitosamente a su destino. Por lo tanto existe la posibilidad de que si ocurre una falla después de enviarlos a su destino pero antes de reconocerlos, puede que sean enviados de nuevo cuando el sistema se recupere.

Este modo está disponible para mensajes persistentes y no-persistentes

`QOS_ONCE_AND_ONLY_ONCE`

Este modo especifica que los mensajes llegarán exactamente una vez. Cuando la fuente del mensaje y el destino se encuentran en la misma instancia del servidor de JBoss Messaging, el mensaje se puede enviar y recibir en la misma transacción local.

Si la fuente y el destino se encuentran en servidores diferentes, puede implementar una alta

durabilidad del mensaje utilizando una transacción JTA controlada por la implementación JTA de JBoss Transactions. Si se requiere JTA, ambas fábricas de conexiones deben ser implementaciones **`XAConnectionFactory`**.

Este modo solo está disponible para mensajes persistentes

Este modo requiere el registro en el administrador de transacciones y en el lado del recurso para soportar la recuperación. Si requiere este nivel de calidad del servicio entonces tiene que habilitar la recuperación XA con JBoss Transactions.

Métodos opcionales

YEs posible que pueda aplicar la semántica *una y solo una vez* a una aplicación en especifico sin configurar **`QOS_ONCE_AND_ONLY_ONCE`**. Establezca el modo **`QOS_DUPLICATES_OK`** y luego busque y descarte mensajes duplicados en el destino.
Puede implementar **`QOS_ONCE_AND_ONLY_ONCE`** a nivel de la aplicación manteniendo un caché de IDs de mensajes recibidos en el disco y comparando los mensajes recibidos en este caché. El caché sólo será válido por un periodo de tiempo así que este enfoque no es tan infalible pero puede ser una opción útil dependiendo de su aplicación.

8.2. Despliegue del puente

Puede implementar un puente de mensajes agregando un descriptor bean administrado en el directorio **`deploy`** de su instalación de JBoss Enterprise Application Platform installation, el cual contiene JBoss Messaging

8.3. Configuración del puente

El siguiente código es un ejemplo de la configuración del puente de mensajes con todos los atributos. A algunos atributos se les quitó el comentario para esta configuración ya que no es apropiado especificarlos todos al mismo tiempo. Los que debe especificar depende de la configuración que quiera.

Ejemplo 8.1. Configuración del puente de mensajes

```
<mbean code="org.jboss.jms.server.bridge.BridgeService"
  name="jboss.messaging:service=Bridge,name=TestBridge"
  xmbean-dd="xmdesc/Bridge-xmbean.xml">

  <!-- The JMS provider loader that is used to lookup the source
       destination
  -->
  <depends optional-attribute-name="SourceProviderLoader">
  jboss.messaging:service=JMSProviderLoader,name=JMSProvider</depends>

  <!-- The JMS provider loader that is used to lookup the target
       destination
  -->
  <depends optional-attribute-name="TargetProviderLoader">
  jboss.messaging:service=JMSProviderLoader,name=JMSProvider</depends>

  <!-- The JNDI lookup for the source destination -->
  <attribute name="SourceDestinationLookup">/queue/A</attribute>

  <!-- The JNDI lookup for the target destination -->
  <attribute name="TargetDestinationLookup">/queue/B</attribute>

  <!-- The username to use for the source connection
  <attribute name="SourceUsername">bob</attribute>
  -->

  <!-- The password to use for the source connection
  <attribute name="SourcePassword">BobSecur3</attribute>
  -->

  <!-- The username to use for the target connection
  <attribute name="TargetUsername">mary</attribute>
  -->

  <!-- The password to use for the target connection
  <attribute name="TargetPassword">MaryS3cur3</attribute>
  -->

  <!-- Optional: The Quality Of Service mode to use, one of:
    QOS_AT_MOST_ONCE = 0;
    QOS_DUPLICATES_OK = 1;
    QOS_ONCE_AND_ONLY_ONCE = 2;
  -->
  <attribute name="QualityOfServiceMode">0</attribute>

  <!-- JMS selector to use for consuming messages from the source
  <attribute name="Selector">specify jms selector here</attribute>
  -->

  <!-- The maximum number of messages to consume from the source
       before sending to the target
  -->
  <attribute name="MaxBatchSize">5</attribute>

  <!-- The maximum time to wait (in ms) before sending a batch to the
       target even if MaxBatchSize is not exceeded. -1 means wait forever
  -->
  <attribute name="MaxBatchTime">-1</attribute>

  <!-- If consuming from a durable subscription this is the subscription
       name
       <attribute name="SubName">mysub</attribute>
  -->

  <!-- If consuming from a durable subscription this is the client ID to
       use
       <attribute name="ClientID">myClientID</attribute>
  -->
```

```
   <!-- The number of ms to wait between connection retrues in the event
        connections to source or target fail
   -->
   <attribute name="FailureRetryInterval">5000</attribute>

   <!-- The maximum number of connection retries to make in case of failure,
        before giving up -1 means try forever
   -->
   <attribute name="MaxRetries">-1</attribute>

   <!-- If true then the message ID of the message before bridging will be
        added as a header to the message so it is available to the
        receiver. Can then be sent as correlation ID to correlate in a
        distributed request-response
   -->
   <attribute name="AddMessageIDInHeader">false</attribute>

     </mbean>
```

Atributos de configuración del puente de mensajes

SourceProviderLoader, Cargador TargetProvider

El puente utiliza el bean administrado **`JMSProviderLoader`** para buscar la fábrica de conexiones fuente y el destino fuente. Por defecto, JBoss Enterprise Application Platform se envía junto con un **`JMSProviderLoader`**, el cual se implementa en el archivo **`$JBOSS_HOME/server/$PROFILE/deploy/messaging/jms-ds.xml`** y sirve como el **`JMSProviderLoader`** local predeterminado. Para una configuración en clúster, **`hajndi-jms-ds.xml`** realiza la misma tarea.

Si su destino objetivo o fuente se encuentra en un servidor diferente o si corresponde a un proveedor diferente que no es JBoss JMS entonces puede desplegar otra instancia bean administrada **`JMSProviderLoader`** que el puente puede utilizar para contactar el destino en el proveedor JMS remoto.

Para utilizar una fuente o un destino remoto que no es JBoss Messaging con una entrega **`QOS_ONCE_AND_ONLY_ONCE`**, el proveedor JMS remoto debe proporcionar una implementación de recurso JMS XA completamente funcional que funcione remotamente desde el servidor.

SourceDestinationLookup

La búsqueda JNDI completa para el destino fuente por medio del **`SourceProviderLoader`** tal como **`/queue/mySourceQueue`**.

TargetDestinationLookup

La búsqueda JNDI completa para el destino objetivo por medio del **`TargetProviderLocator`** tal como **`/topic/myTargetTopic`**.

SourceUsername

Un atributo opcional que especifica el nombre de usuario para crear la conexión fuente.

SourcePassword

Un atributo opcional que especifica la contraseña para crear la conexión fuente.

TargetUsername

Un atributo opcional que especifica el nombre de usuario para crear la conexión destino.

TargetPassword

Un atributo opcional que especifica la contraseña para crear la conexión destino.

QualityOfServiceMode

Un número entero que representa el modo de la *calidad de servicio* deseada. Los valores posibles son:

- **0** que representa **QOS_AT_MOST_ONCE**
- **1** que representa **QOS_DUPLICATES_OK**
- **2** que representa **QOS_ONCE_AND_ONLY_ONCE**

Consulte la Sección 8.1, “Sinopsis del puente de mensajes ” para obtener una explicación completa de estos modos.

Selector

Un atributo opcional que le permite proporcionar una expresión selector JMS al consumir mensajes desde un destino fuente. Sólo los mensajes que coincidan con la expresión del selector serán transportadas a través de un puente desde la fuente hasta el destino. La expresión del selector debe seguir la sintaxis del selector JMS especificada aquí: http://java.sun.com/j2ee/1.4/docs/api/javax/jms/Message.html.

Para mayor rendimiento, aplique los selectores de subscripción de temas fuente a consumidores de colas fuente.

MaxBatchSize

Especifica el número máximo de mensajes a consumir desde el destino fuente antes de enviarlos en grupo al destino objetivo. Su valor debe sermayor o igual a **1**.

MaxBatchTime

Especifica el número máximo de milisegundos a esperar antes de enviar un grupo de mensajes al destino incluso si no llegado hasta **MaxBatchSize**. Su valor dede ser **-1** (espere por siempre) o mayor o igual a **1** para especificar un tiempo.

SubName

Representa el nombre de la suscripción durable que consumirá del tema destino fuente.

ClientID

Representa el ID de cliente JMS a utilizar al crear o buscar la suscripción durable que consumirá del tema destino fuente.

FailureRetryInterval

La cantidad de tiempo (en milisegundos) a esperar entre el tratar de recrear las conexiones a los servidores destino o fuente después de que se detecta un fallo.

MaxRetries

El número de veces a intentar recrear las conexiones a los servidores fuente o destino después de detectar fallos. El puente parará de intentar recrear la conexión. Un valor de **-1** significa que el puente tratará de reconectarse indefinidamente.

AddMessageIDInHeader

Cuando es **true** el id del mensage original se añadirá en el encabezado de **JBossMessage.JBOSS_MESSAGING_BRIDGE_MESSAGE_ID_LIST** del mensaje que se está enviando al destino. Si se realiza un puente para el mensaje más de una vez entonces se añadirá el de ID de cada mensaje al encabezado. Esto activa un patrón de petición-respuesta

distribuído.

Capítulo 9. Habilitación del grupo de ordenamiento de JBoss Messaging

Esta sección describe la manera de utilizar la funcionalidad del grupo de ordenamiento de JBoss Messaging para lograr un ordenamiento de mensajes estricto.

Los grupos de ordenamiento de mensajes es la implementación de JBoss Messaging del ordenamiento estricto de mensajes. Cuando se habilita la funcionalidad del grupo de ordenamiento, las prioridades del mensaje ya no influencian el orden en que los mensajes se entregan. Los mensajes en un grupo de ordenamiento particular se entregarán en el orden exacto en que llegan a la cola destino.

Los grupos de ordenamiento están identificados por sus nombres en cadena y siguen las siguientes reglas:

Regla uno

Los mensajes que forman parte de un grupo de ordenamiento se entregan uno a la vez. El siguiente mensaje no se entregará hasta que se complete la entrega de un mensaje anterior. Hay varias maneras de señalar que la entrega de un mensaje se ha completado, dependiendo de la configuración del modo de reconocimiento.

- El modo **`CLIENT_ACKNOWLEDGE`** hace que el método **`Message.acknowledge()`** señale el estado de completo.
- Los modos **`AUTO_ACKNOWLEDGE`** y **`DUPS_OK_ACKNOWLEDGE`** hacen que al completarse el mensaje se identifique con uno de los siguientes:
  - un retorno exitoso de uno de los métodos **`MessageConsumer.receive()`** o
  - un retorno exitoso de la llamada **`onMessage()`** del **`MessageListener()`**.

Nota

Si el consumidor de mensajes está cerrado entonces se considerará como completado el mensaje que se está procesando en el momento del cierre. Esto es sin importar si se ha llamado a un **`*_ACKNOWLEGE`** antes del cierre del consumidor.

Regla dos

En el caso de recepción transaccional de mensajes, el siguiente mensaje no se entregará hasta que la transacción haya guardado los cambios, lo cual incluye el reconocimiento de la recepción del mensaje actual. Si la transacción se deshace entonces el mensaje se cancelará, se enviará de regreso al servidor JMS y se hará disponible para la siguiente entrega.

9.1. Cómo habilitar el grupo de ordenamiento de mensajes

El grupo de ordenamiento de JBoss Messaging se puede habilitar de dos maneras. Ya sea utilinzado el API o realizando cambios en la configuración.

9.1.1. Habilitación con el API

Para utilizar la funcionalidad del grupo de ordenamiento de JBoss Messaging es necesario crear un **`JBossMessageProducer`** como se demuestra en el siguiente ejemplo del código:

```
JBossMessageProducer producer=(JBossMessageProducer)session.createProducer(queue);
```

Una vez creado, JBossMessageProducer proporciona dos métodos para iniciar y terminar un grupo de ordenamiento.

enableOrderingGroup()

El siguiente ejemplo de código demuestra la manera de crear un grupo de ordenamiento con el nombre

ogrpName.

```
public void enableOrderingGroup(String ogrpName) throws JMSException
```

Una vez que se llame, el productor enviará mensajes de parte del grupo de ordenamiento. Si se proporciona un parámetro **null** entonces se generará de manera automática el nombre del grupo de ordenamiento. El llamar este método más de una vez sobreescribirá cualquier llamada anteriro del método.

disableOrderingGroup()

El siguiente ejemplo de código demuestra cómo parar de producir mensajes de grupo de ordenamiento.

```
public void disableOrderingGroup() throws JMSException
```

Una vez que se llame, el productor parará de enviar mensajes del grupo de ordenamiento y re-iniciará su comportamiento predeterminado.

9.1.2. Cambios en la configuración

Los usuarios pueden configurar una fábrica de conexiones JBoss Messaging para habilitar la funcionalidad del grupo de ordenamiento. Para lograr esto, se agregan dos nuevos atributos al archivo de configuración del servicio de fábrica. Estos son:

- **EnableOrderingGroup;**
  - configure esta propiedad como **true** para habilitar la funcionalidad del grupo de ordenamiento. El valor predeterminado para esta propiedad es **false**.
- **DefaultOrderingGroupName;**
  - el nombre predeterminado para el grupo de ordenamiento de los mensajes. El nombre del grupo se generará automáticamente si este atributo falta.

Nota

Una vez configurado para habilitar la funcionalidad del grupo de ordenamiento en una fábrica de conexiones, todos los mensajes que se envían desde cualquier productor creado desde la fábrica de conexiones se convierten en mensajes del grupo de ordenamiento.

El siguiente ejemplo del archivo de configuración del servicio de fábrica demuestra cómo habilitar la funcionalidad del grupo de ordenamiento:

```
<mbean code="org.jboss.jms.server.connectionfactory.ConnectionFactory";
  name="jboss.messaging.connectionfactory:service=ConnectionFactory";
  xmbean-dd="xmdesc/ConnectionFactory-xmbean.xml">
  <depends optional-attribute-name="ServerPeer">
    jboss.messaging:service=ServerPeer
  </depends>
  <depends optional-attribute-name="Connector">
    jboss.messaging:service=Connector,transport=bisocket
  </depends>
  <depends>
    jboss.messaging:service=PostOffice
  </depends>

  <attribute name="JNDIBindings">
    <bindings>
    <binding>/MyConnectionFactory</binding>
    <binding>/XAConnectionFactory</binding>
    <binding>java:/MyConnectionFactory</binding>
    <binding>java:/XAConnectionFactory</binding>
    </bindings>
  </attribute>

  <!-- This are the two properties -->
  <attribute name="EnableOrderingGroup">true</attribute>
  <attribute name="DefaultOrderingGroupName">MyOrderingGroup</attribute>
</mbean>
```

La ventaja de habilitar la funcionalidad del grupo de ordenamiento al realizar cambios en la configuración es la facilidad con la que se puede lograr la funcionalidad de ordenamiento de mensajes sin la necesidad de cambiar el código.

9.2. Notas y limitaciones

Debe notar los siguientes puntos con relación a la funcionalidad del grupo de ordenamiento:

- Las colas se tienen que utilizar con la funcionalidad del grupo de ordenamiento. La funcionalidad no servirá con temas.
- La funcionalidad del grupo de ordenamiento no se debe utilizar junto con selectores de mensajes ni la entrega programada.
- Un mensaje se considera completo y el siguiente mensaje estará disponible para la entrega, si el mensaje original está muerto o si ha caducado. Un mensaje muerto se mueve al **`DLQ`** mientras que un mensaje caducado se mueve a la **`ExpiryQueue`**.
- Al utilizar un **`ConnectionConsumer`**, se observará el ordenamiento de los mensajes. Sin embargo, el **`ConnectionConsumer`** no controla la sesión que recibirá el siguiente mensaje.
- En el caso de una cola distribuída, el usuario debe utilizar **`HASingleton`** para asegurarse de que la funcionalidad del grupo de ordenamiento sirve de manera correcta.

Historial de revisiones

Revisión 5.1.0-2.400	**2013-10-31**	**Rüdiger Landmann**

Rebuild with publican 4.0.0

Revisión 5.1.0-2	**2012-07-18**	**Anthony Towns**

Rebuild for Publican 3.0

Revisión 5-1	**Wed Sep 15 2010**	**Jared Morgan**

Se cambió el número de la versión de acuerdo con los nuevos requerimientos de la versión.
Revisado para JBoss Enterprise Application Platform 5.1.0.GA, incluyendo:
Se modificaron los capítulos 6 y 7 para el proyecto de criterios comunes 5.1. Se resaltó el código donde era apropiado.
JBOSSCC-49 - se incorporaron todos los comentarios del nuevo manual de JBoss Messaging.